DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	9/22/93	Final Technical 9/15/91-10/31/93

4. TITLE AND SUBTITLE
Development and Application of New Algorithms for the Simulation of Viscouse Scompressible Flows with Moving Bodies in Three Dimensions

5. FUNDING NUMBERS
F49620-92-J-0058
2307/A5

6. AUTHOR(S)
Rainold Lohner, Jean Cabello

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Office of Sponsored Research
The George Washington University
2121 I Street, N.W., Suite 601
Washington, DC 20052

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 93 0861

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NA
AFOSR/
Department of the Air Force
Air Force Office of Scientific Research, Bolling

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-92-J-0058

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The work carried out under this contract may be subdivided according to the following topics: 1) Development of Implicit ALE Navier-Stokes Solvers;
2) Implementation of Turbulence Models;
3) Development of Gridding Techniques for 3-D Viscous Flows;
4) Demonstration Calculation and Results.

14. SUBJECT TERMS	15. NUMBER OF PAGES
TURBULENCE, VISCOUS FLOWS, NAVIER-STOKES SOLVERS	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
U	U	U	

Final Report for Contract AFOSR-F-49620-92-J0058

Contract Monitor: Dr. Leonidas Sakell

Prepared by:

Rainald Löhner and Jean Cabello
CMEE/SEAS, The George Washington University
Washington, D.C. 20052

Accomplishments

The work carried out under this contract may be subdivided according to the following topics: 1) Development of Implicit ALE Navier-Stokes Solvers;
2) Implementation of Turbulence Models;
3) Development of Gridding Techniques for 3-D Viscous Flows;
4) Demonstration Calculation and Results.

The description of the main accomplishments of the present work are listed according to these topics in the following.

1. IMPLICIT ALE NAVIER-STOKES SOLVERS

The work performed in the area of flow solvers may be subdivided into two parts:
a) Improvement of Edge-Based Solvers;
b) Implicit Solvers for the Navier-Stokes equations.

1.1 Edge-Based Solvers: The switch from element-based to edge-based solvers allowed a number of improvements in the performance and accuracy of the flow-codes. When developing these new codes, we incorporated all the coding lessons that we learned over the years. The latest code, FEFLO96, runs at a sustained rate of 115 MFlops on the CRAY-YMP, and has significantly less Flops per update than the previous code (FEFLO52). At the same time, indirect addressing (i/a) costs were reduced by a factor of 7.3. This reduction in i/a was achieved by using the edge-based data structure (1.57), computing the fluxes 'on the fly' from the unknowns (2.14) and using superedges (2.18) [F1]. The code without superedges runs about 25% faster on the CRAY-YMP than the code using usual edges. We also excercised FEFLO96 in parallel on the CRAY-C90, and simply using autotasking, i.e. CRAY-preprocessing, timed a remarkable 3.87 speed-up on 4 processors for some large runs.

1.2 Implicit Solvers: The use of implicit solvers was recognized as being essential for the fast solution of the compressible Navier-Stokes equations. Therefore, we developed implicit iterative solvers. The solvers are based on a linearized Euler backward time-advancement of the Navier-Stokes equations. The resulting non-linear system of equations is then solved using a preconditioned GMRES solver. Both block-diagonal and incomplete LU decompositions were tried. While incomplete LU performs better than block-diagonal, it also requires a lot more memory, and is less amenable to massive parallel architectures. For this reason, we are currently investigating other preconditioners. More details of these techniques are given in [4], which is included in the current report as Appendix 1.

2. TURBULENCE MODELS

The algebraic Baldwin-Lomax turbulence model [T1] was implemented in the 3-D flow solver. This is the simplest way to add the effects of turbulence. The model is straightforward to implement as the only difference in the flow equations is the increase of viscosity. The model represents the inner and outer parts of a boundary layer as follows:

Inner part:

$$\mu_t = \rho|\omega|\,[ky(1 - exp(-y/A))]^2$$

Outer part:

$$\mu_t = \alpha C_{cp}\rho Y_{max} F_{max}\gamma \quad ,$$

with

$$F(y) = y|\omega|(1 - exp(-y/A)) \quad .$$

The exponential factor in the inner part is the Van Driest damping factor which matches the damping of the wall. In the second equation, F_{max} is computed along a normal to the wall. This is particularly easy when dealing with structured grids, but requires some effort for unstructured grids. The required data structures were discussed by Rostand [T2] for 2-D. As far as we can see, this is the first time a fully unstructured 3-D Baldwin-Lomax implementation has been attempted. One complete evaluation of the turbulent viscosity requires the following transfer of information:
a) Vorticity from elements or points in the mesh to the appropriate normals to the walls. Along each normal, the vorticity $|\omega|$ is required to evaluate the turbulent viscosity. This transfer of vorticity is accomplished with a linked list of intersections of normals to the wall with elements of the mesh. This list is constructed by starting from the surface and moving along the normal. All that is required to do so is a list of elements surrounding elements.

b) Transfer of the turbulent viscosity from the normals to the walls to the elements or points of the mesh. In the present case, we transfer to points. This not only reduces the transfers required (there are less points than elements in a mesh), but also introduces a beneficial smoothing effect at element level. For each point in the mesh close to wetted surfaces, we find the closest normals surrounding it, and then the two closest points along each of the normals. Thus, a point in the mesh assembles the turbulent viscosity from four points along the normals of a wetted surface. The search for the closest points is done using quad-trees [T3]. In the case of the mixing of several boundary layers, μ_t is computed from the contribution of each wall weighted by its distance to the point.

3. GRIDDING FOR 3-D VISCOUS FLOWS

The difficulty of gridding complex geometries for the simulation of flows using the Navier-Stokes equations - i.e. including the effects of viscosity and the associated boundary or mixing layers - increases not only with the geometric complexity of the domain to be gridded, but also with the Reynolds-number of the flow. For high Reynolds-numbers, the proper discretization of the very thin, yet important boundary or mixing layers requires elements with aspect ratios well in excess of 1:1000. This requirement presents formidable difficulties to general, 'black-box' unstructured grid generators. These difficulties can be grouped into two main categories:

a) Amount of Manual Input: In most unstructured grid generators, the desired spatial distribution of element size and shape is given by some form of background grid or sources [G1,G2]. This seems natural within an adaptive context, as a given grid, combined with a suitable error indicator/estimator, can then be used as a background grid to generate an even better grid for the problem at hand. Consider now trying to generate from manual input a first grid that achieves stretching ratios in excess of 1:1,000. The amount of background gridpoints or sources required will be proportional to the curvature of the objects immersed in the flowfield. This implies an enormous amount of manual labor for general geometries, rendering this approach impractical.

b) Loss of Control: Most unstructured grid generators introduce a point or element at a time, checking the surrounding neighbourhood for compatibility. These checks involve Jacobians of elements and their inverses, distance-functions, and other geometrical operations that involve multiple products of coordinate differences. It is not difficult to see that as the stretching-ratio increases, round-off errors can become a problem. For a domain spanning 1,000m (mesh around a C-17), with a minimum element length at the wing of 0.0001m across the boundary layer and 0.05m along the boundary layer, and a maximum element length of 20m in the farfield, the ratio of element volumes is of the order of $3 * 10^{-11}$. Althoughthis is well within reach of the 10^{-16}-accuracy of 64-bit arithmetic, element distortion and surface singularities, as well as loss of control of element shape can quickly push this ratio to the limit.

A number of semi-automatic grid generators have been devised in the past. The most common way to generate meshes suitable for Navier-Stokes calculations for complex

geometries is to employ a structured or semi-structuredmesh close to wetted surfaces or wakes [G3-G10]. This 'Navier-Stokes' region mesh is then linked to an outer unstructured grid that covers the 'inviscid' regions. In this way, the geometric complexity is solved using unstructured grids and the physical complexity of near-wall or wake regions is solved by semi-structured grids. This approach has proven very powerful in the past, as evidenced by many examples.

A recurring problem in all of these approaches has been how to link the semi-structured mesh region with the unstructured mesh region. We developed a new, general technique to solve this problem. The design criteria for the new grid generation strategy may be summarized as follows:

- The geometric flexibility of the unstructured grid generator should not be compromised for Navier-Stokes meshes. This implies using unstructured grids for the surface discretization.
- The manual input required for a desired Navier-Stokes mesh should be as low as that used for the Euler case. In the present case, this requirement is solved by specifying at the points of the background grid the boundary layer thickness and the geometric progression normal to the surface.
- The generation of the semi-structured grid should be fast. Experience shows that usually more than half of the elements of a typical Navier-Stokes mesh are located in the boundary-layer regions. This requirement is met by constructing the semi-structured grids with the same normals as encountered on the surface, i.e. without recurring to smoothing procedures as the semi-structured mesh is advanced into the field [G9,G11].
- The element size and shape should vary smoothly when going from the semi-structured to the fully unstructured mesh regions. How to accomplish this is the main topic of this paper, and is detailed in subsequent sections.
- The grid generation procedure should avoid all of the problems typically associated with the generation of Navier-Stokes meshes for regions with high surface curvature: negative or deformed elements due to converging normals, and elements that get too large due to diverging normals at the surface. In order to circumvent these problems, the same techniques which are used to achieve a smooth matching of semi-structured and unstructured mesh regions are used.

Given these design criteria, as well as the approaches used to meet them, the present grid generation algorithm can be summarized as follows:

M.1 Given a surface definition and a background grid, generate a surface triangulation using an unstructured grid generator.

M.2 From the surface triangulation, obtain the surface normals.

M.3 Smooth the surface normals in several passes in order to obtain a more uniform mesh in regions with high surface curvature.

M.4 Construct a semi-structured grid with the information provided by the background grid and the smoothed normals.

M.5 Examine each element in this semi-structured region for size and shape; remove all elements that do not meet certain specified quality criteria.

M.6 Examine whether elements in this semi-structured region cross each other; if so, keep the smaller elements and remove the larger ones, until no crossing occurs.

M.6 Examine whether elements in this semi-structured region cross boundaries; if so, remove the crossing elements.

M.7 Mesh the as yet 'empty' regions of the computational domain using the background grid and the unstructured grid generator.

The main areas of work were:

a) Smoothing of Surface Normals: This implies obtaining the smoothed normals quickly (i.e. less than 20 passes over the surface mesh), and defining proper boundary conditions for the normals in order to avoid problems at ridges, intersections, etc.

b) Prism Generation: When generating tetrahedra from prisms, certain compatibility criteria must be met [G14]. We developed a very fast compatibility algorithm that was found to converge in less than 3 passes over the surface mesh.

c) Fast Proximity Finders for Crossed Elements/Points: In order to speed up the checking of bad elements (negative, crossing, etc.), a series of data structures had to be developed and implemented. The main data structures used are Octrees and Linked Lists. The main filtering strategies to reduce the work even further are cones of visibility and distance functions.

d) Initial Front for Unstructured Grid Region: Once the elements have been marked for deletion, the initial front for the unstructured grid generation of the remaining 'empty' region of space has to be obtained. The main work here was to obtain this initial front taking into consideration the boundary arrays required by the flow-solver later on, i.e. to minimize user-intervention as much as possible.

More details of the algorithm, as well as several example grids computed with it, are given in [3], which is included in the report as Appendix 2.

4. DEMONSTRATION CALCULATIONS AND RESULTS

4.1 Unstructured Grid/Remeshing Transient 3-D Runs: With the developed tools we performed several 2-D and 3-D test runs. The steady 2-D and 3-D results are summarized in [1], which is included here as Appendix 3. Pilot/Seat ejection from an F-16 fighter at supersonic speeds was shown in [2], which is included here as Appendix 4.

5. PUBLICATIONS

All of the developments listed above were reported extensively in the literature. The main papers published are listed in chronological order:

[1] H. Luo, J.D. Baum, R. Löhner and J. Cabello - Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations; *AIAA*-93-0336 (1993).

[2] J.D. Baum and R. Löhner - Numerical Simulation of Pilot/Seat Ejection from an F-16; *AIAA*-93-0783 (1993).

[3] R. Löhner - Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations; *AIAA*-93-3348-CP (1993).

[4] H. Luo, J.D. Baum, R. Löhner and J. Cabello - An Implicit Three-Dimensional Finite Element Solver for Unstructured Meshes; pp. 1027,1028 in *Proc. 11th AIAA CFD Conf.* , Orlando, FL, July (1993).

[5] H. Luo, J.D. Baum, R. Löhner- Numerical Solution of the Euler Equations for Complex Aerodynamic Configurations Using An Edge-Based Finite Element Scheme; *AIAA*-93-2933 (1993).

6. CONCLUSIONS AND OUTLOOK

We have made major steps towards the development of a CFD capability to compute viscous 3-D compressible flows with moving bodies.

In the future, we plan to expand the developed capabilities as follows:

a) Development of Flow Solvers for high Re-number Viscous Flows: We will continue to develop our implicit Navier-Stokes solvers. In particular, we plan to extend the linelet-concept to the fully coupled linear equation systems that arise for implicit Navier-Stokes solvers.

b) Turbulence Models: We will incorporate the $k - \epsilon$ model into the implicit Navier-Stokes solver.

c) Development of suitable gridding algorithms for high Re-number viscous flows: We plan to improve our present capability for automatically gridding problems involving high Re-number viscous flows.

d) Development of suitable error indicators for high Re-number viscous flows: The idea here is to develop error indicators that sense were to refine boundary layers and shear layers, and that work even for highly stretched grids.

e) Improvements in movie-making capabilities: The main aim of this improvement is to be able to run the movie on the workstation before going to the movie-making center. At the same time, we must be able to compress the information to an extent that a 3min movie can be stored effortlessly on a small disk. This will be accomplished through image compression algorithms.

f) Further test runs: we plan to look for available experimental and/or numerical data in the literature in order to set up runs to test the algorithms developed.

7. REFERENCES

Flow Solvers:

[F1] R. Löhner - Edges, Stars, Superedges and Chains; GWU-CMEE Rep. 91/92-01, submitted to *Comp. Meth. Appl. Mech. Eng.* (1992).

Turbulence Models:

[T1] B.S. Baldwin and H. Lomax - Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows AIAA 78-257 (1978).

[T2] P. Rostand - Algebraic Turbulence Models for the Computation of 2-D High Speed Flows Using Unstructured Grids; *ICASE Rep.* 88-63 (1988).

[T3] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).

Grid Generation:

[G1] P.L. George - *Automatic Mesh Generation*; J. Wiley & Sons (1991).

[G2] W. Dekonick and T. Barth (eds.) - *AGARD Rep.* 787, *Proc. Special Course on Unstructured Grid Methods for Advection Dominated Flows*, VKI, Belgium, May (1992), Chapter 8.

[G3] K. Nakahashi - FDM-FEM Zonal Approach for Viscous Flow Computations over Multiple Bodies; *AIAA*-87-0604 (1987).

[G4] K. Nakahashi and S. Obayashi - Viscous Flow Computations Using a Composite Grid; *AIAA-CP*-87-1128, 8th CFD Conf., Hawaii (1987).

[G5] J.F. Thompson - A Composite Grid Generation Code for General 3D Regions - The EAGLE Code; *AIAA J.* **26, 3**, 271ff (1988).

[G6] J.P. Steinbrenner, J.R. Chawner and C.L. Fouts - A Structured Approach to Interactive Multiple Block Grid Generation; *AGARD-CP-464*, 8 (1990).

[G7] S. Allwright - Multiblock Topology Specification and Grid Generation for Complete Aircraft Configurations; *AGARD-CP-464*, 11 (1990).

[G8] F.C. Dougherty and J. Kuan - Transonic Store Separation Using a Three-Dimensional Chimera Grid Scheme; *AIAA*-89-0637 (1989).

[G9] K. Nakahashi - Optimum Spacing Control of the Marching Grid Generation; *AIAA*-88-0515 (1988).

[G10] R. Ramamurti and R. Löhner - Simulation of Subsonic Viscous Flows Using Unstructured Grids and a Finite Element Solver;*AIAA*-90-0702 (1990).

[G11] Y. Kallinderis and S. Ward - Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations; *AIAA*-92-2721 (1992).

[G12] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* **4**, 123-135 (1988).

[G13] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* **8**, 1135-1149 (1988).

[G14] R. Löhner - Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations; AIAA-93-3348-CP (1993).

APPENDIX 1: IMPLICIT SOLVERS

AN IMPLICIT THREE-DIMENSIONAL FINITE ELEMENT SOLVER FOR UNSTRUCTURED MESHES

Hong Luo[1], Joseph D. Baum[1], Rainald Löhner[2], and Jean Cabello[2]

[1]Science Applications International Corporation
1710 Goodridge Drive, MS 2-3-1, McLean, VA 22102
[2]CMEE, School of Engineering and Applied Science
The George Washington University, Washington, D.C. 20052

INTRODUCTION

Significant progress has been made in recent years in developing numerical algorithms for the solution of the compressible Euler and Navier-Stokes equations on unstructured grids. Most efforts have been focused on improving the spatial discretization operator which has reached a high degree of sophistication. Usually, explicit time integration, such as the multi-stage Runge-Kutta scheme has been used to get a steady state solution. In general, explicit schemes are easy to implement and vectorize and require only limited memory storage. However, for large-scale problems and especially for solution of the Navier-Stokes equations, the rate of convergence slows dramatically. To speed up the convergence rate, an implicit temporal discretization is required.

The objective of this research is to develop an implicit 3D finite element algorithm for the solution of the compressible Euler and Navier-Stokes equations on unstructured meshes. Numerical results for both inviscid and viscous flows are presented to demonstrate the performance of the proposed scheme.

NUMERICAL SCHEME

The governing equations are integrated in time using an Euler implicit differencing scheme. The resulting large nonsymmetric linear system of equations is solved by using the preconditioned GMRES algorithm. The preconditioner used in this work is a block incomplete LU factorization with zero fill-in. Left, right, and symmetric preconditioners are investigated and examined. Spatial discretization is achieved by using an edge-based finite element scheme [1]. Roe's flux-difference splitting is used for spatial discretization of the inviscid flux terms. A MUSCL approach is used to achieve higher-order accuracy. The viscous flux terms are evaluated using second order accurate central differences.

Results obtained during this investigation indicate that the treatment of boundary conditions is extremely important to the success of an implicit scheme. When boundary conditions are treated explicitly, only a very limited CFL number can be used. In order for the implicit scheme to be stable for high CFL numbers, the boundary condition must be treated implicitly. In the present work, Roe's approximate Riemann solver was also extended to treat the boundary points similarly to the interior points. This procedure provides a boundary point treatment that is completely compatible and consistent with the interior point differencing scheme. The boundary conditions are then linearized consistently, and are included in the left-hand-side coefficient matrix.

NUMERICAL RESULTS

Due to space limitation, results are presented only for an inviscid transonic flow and a laminar viscous flow. The first test case is the well known Ni's test case. It is an inviscid flow in a channel with a 10% thick circular bump on the bottom. Inlet Mach number is 0.675. This is a 3D simulation of a 2D flow. The mesh, which contains 13,891 grid points, 68,097 element and 4,442 boundary points, is depicted in Fig.1a. Fig.1b displays the computed pressure contours in the flow field. The Mach number distribution on lower wall is shown in Fig.1c. Fig.1d displays a comparison of convergence histories between explicit scheme and implicit scheme with left, right, and symmetric ILU preconditioner, respectively. The explicit scheme results were obtained using three stage Runge-Kutta scheme with implicit residual smoothing and a CFL number of 4. The implicit scheme results were obtained using a CFL number of 100,000 with a maximum number of Krylov space of 10 without restart. Contrary to the results obtaind in [2], the left, right and symmetric preconditioners perform equally well.

The second test case involves a 3D simulation of a 2D laminar flow past a flat plate at a Mach number of 0.5 and a chord Reynolds number of 10,000. The mesh, shown in Fig.2a. contains 81,885 elements, 15,694 points, and 3,774 boundary points. The computed Mach number contours and velocity vectors in the flow field are depicted in Fig.2b and 2c. Fig.2d shows the comparison of the Blasius velocity profile and the computed velocity profiles as scaled by the Blasius similarity law at the exit.

REFERENCES

[1] H. Luo, J. D. Baum, R. Löhner and J. Cabello - Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations on Unstructured meshes; AIAA-93-0336.

[2] V. Venkatakrishnan - Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations; AIAA-90-0586.

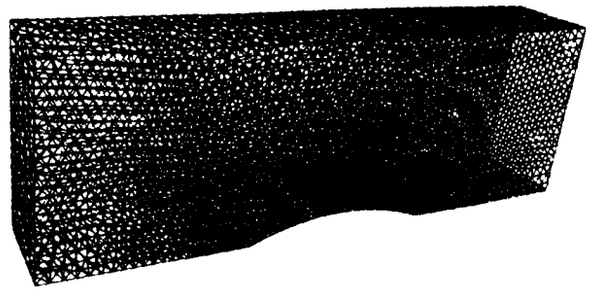


Fig.1a Mesh in a channel

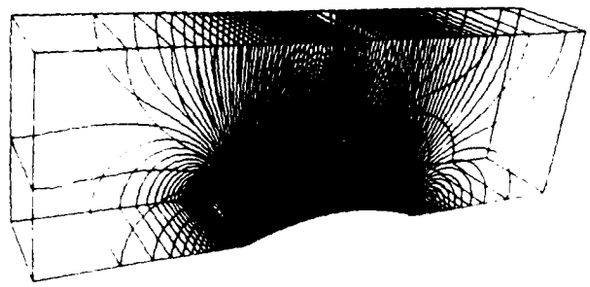


Fig.1b Pressure contours

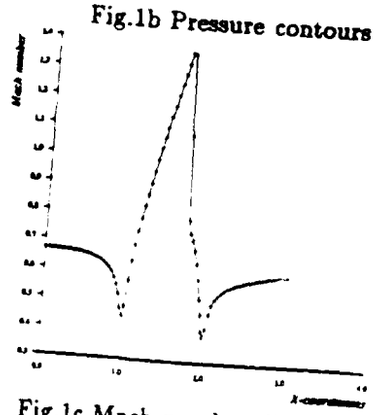


Fig.1c Mach number distribution

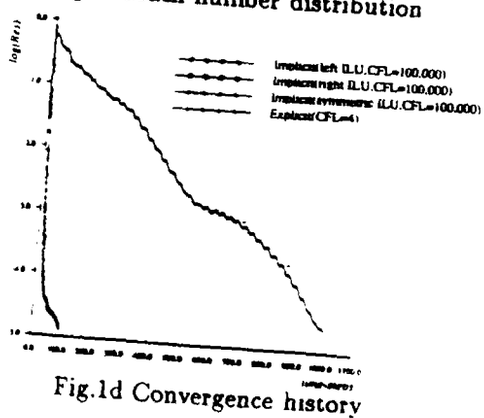


Fig.1d Convergence history

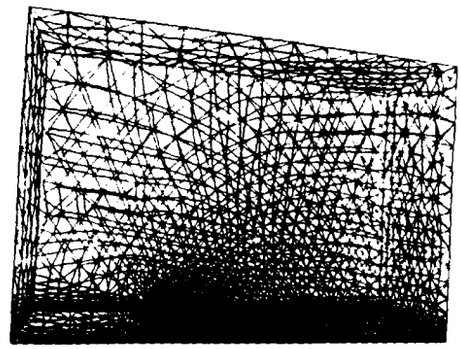


Fig.2a Mesh for a flat plate

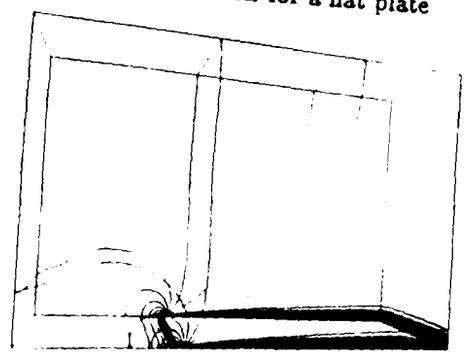


Fig.2b Mach number contours

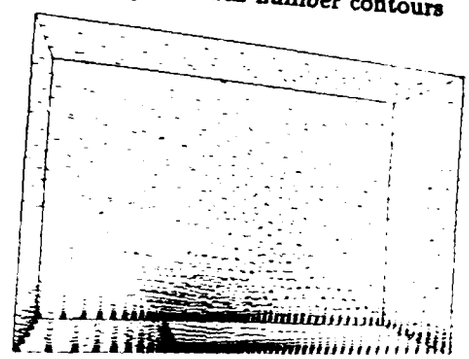


Fig.2c Velocity vectors

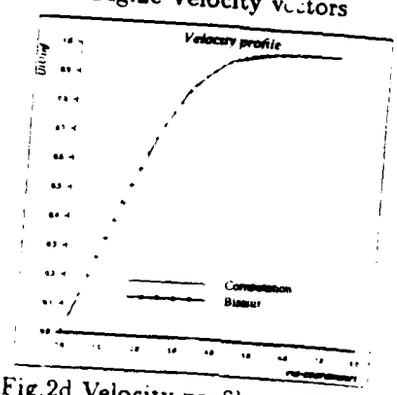


Fig.2d Velocity profile vs. Blasius

APPENDIX 2: NAVIER-STOKES GRIDDING TECHNIQUE

AIAA-93-3348-CP

MATCHING SEMI-STRUCTURED AND UNSTRUCTURED GRIDS FOR NAVIER-STOKES CALCULATIONS

Rainald Löhner

CMEE, School of Engineering and Applied Science
The George Washington University, Washington, D.C. 20052

ABSTRACT

A new gridding technique for Navier-Stokes calculations involving complex geometries is presented. This technique is based on a combination of semi-structured and unstructured meshing techniques that accommodates the strengths of these two approaches while avoiding their respective weaknesses. The technique has the advantage of being generally applicable, yielding one single unstructured final mesh for a given computational domain. At the same time, the problems usually encountered when meshing surfaces with high curvature for Navier-Stokes calculations are avoided automatically.

1. INTRODUCTION

The task of gridding complex geometries for the simulation of flows using the Navier-Stokes equations - i.e. including the effects of viscosity and the associated boundary or mixing layers - is encountered commonly in engineering practice. The difficulty of this task increases not only with the geometric complexity of the domain to be gridded, but also with the Reynolds-number of the flow. For high Reynolds-numbers, the proper discretization of the very thin, yet important boundary or mixing layers requires elements with aspect ratios well in excess of 1:1,000. This requirement presents formidable difficulties to general, 'black-box' unstructured grid generators. These difficulties can be grouped into two main categories:

a) Amount of Manual Input: In most unstructured grid generators, the desired spatial distribution of element size and shape is given by some form of background grid or sources [1,2]. This seems natural within an adaptive context, as a given grid, combined with a suitable error indicator/estimator, can then be used as a background grid to generate an even better grid for the problem at hand. Consider now trying to generate from manual input a first grid that achieves stretching ratios in excess of 1:1,000. The amount of background gridpoints or sources required will be proportional to the curvature of the objects immersed in the flowfield. This implies an enormous amount of manual labor for general geometries, rendering this approach impractical.

b) Loss of Control: Most unstructured grid generators introduce a point or element at a time, checking the surrounding neighbourhood for compatibility. These checks involve Jacobians of elements and their inverses, distance-functions, and other geometrical operations that involve multiple products of coordinate differences. It is not difficult to see that as the stretching-ratio increases, round-off errors can become a problem. For a domain spanning 1,000m (mesh around a Boeing-747), with a minimum element length at the wing of less than 0.01mm across the boundary layer and 0.05m along the boundary layer and along the wing, and a maximum element length of 20m in the farfield, the ratio of element volumes is of the order of $3 * 10^{-12}$. Although this is well within reach of the 10^{-16}-accuracy of 64-bit arithmetic, element distortion and surface singularities, as well as loss of control of element shape can quickly push this ratio to the limit.

Given these difficulties, it is not surprising that at present, there does not exist a 'black-box' unstructured (or structured, for that matter) grid generator that can produce acceptable meshes with such high aspect ratio elements. The demand for Navier-Stokes calculations in or past complex geometries being great, a number of semi-automatic grid generators have been devised. The most common way to generate meshes suitable for Navier-Stokes calculations for complex geometries is to employ a structured or semi-structured mesh close to wetted surfaces or wakes [3-11]. This 'Navier-Stokes' region mesh is then linked to an outer unstructured grid that covers the 'inviscid' regions. In this way, the geometric complexity is solved using unstructured grids and the physical complexity of near-wall or wake regions is solved by semi-structured grids. This approach has proven very powerful in the past, as evidenced by many examples.

The meshes in the semi-structured region can be constructed to be either quads/bricks [3-9] or triangles/prisms [10,11]. The prisms can then be subdivided into tetrahedra if so desired. For the inviscid (unstructured) regions, multi-block approaches have been used for quads/bricks [5-8], and advancing front [13-15], Voronoi [16,17] and modified quadtree/octree approaches [18] for triangles/tetrahedra.

A recurring problem in all of these approaches has been how to link the semi-structured mesh region with the unstructured mesh region. Some solutions put forward have considered:

- Overlapped Structured Grids: these are the so-called chimera grids [8,9], that have become popular for Navier-Stokes calculations of complex geometries; as the gridpoints of the various meshes do not coincide, they allow great flexibility and are easy to construct, but the solution has to be interpolated between grids, which may lead to higher CPU cost and a deterioration in solution quality.
- Overlapped Structured/Unstructured Grids: in this case the overlap zone can be restricted to one cell, with the points coinciding exactly, so that there are no interpolation problems [3,4].
- Delaunay Triangulation of Points Generated by Algebraic Grids: in this case several structured grids are generated, and their spatial mapping functions are stored the resulting cloud of points is then gridded using De

launay triangulation techniques [16].

Although some practical problems have been solved by these approaches, they can not be considered general, as they suffer from the following constraints:

- The first two approaches require a very close link between solver, grid generator and interpolation techniques to achieve good results; from the standpoint of generality, such a close link between solver, grid generator and interpolation modules is undesirable.
- Another problem associated with the first two approaches is that at concave corners, negative (i.e. folded) or badly shaped elements may be generated. The usual recourse is to smooth the mesh repeatedly, or use some other device to introduce ellipticity [10,12]. These approaches tend to be CPU intensive, and require considerable expertise from the user. Therefore, they can not be considered general approaches.
- The third case requires a library of algebraic grids to mesh individual cases, and can therefore not be considered a general tool. However, it has been used extensively for important specialized applications, e.g. single or multi-element airfoil flows [16].

The present research effort is directed towards generality and ease of software expandability and maintainability. Therefore, we strive to generate a single unstructured mesh consisting of triangles/tetrahedra. This mesh can then be considered completely independent of flow solvers, and neither requires any interpolation or other transfer operators between grids, nor the storage of mapping functions.

The remainder of the paper is divided as follows: The design criteria used and the new grid generation strategy are outlined in Section 2. Removal criteria are discussed in Section 3. Section 4 describes the formation of tetrahedra from prismatic elements. Several examples of gridded configurations are shown in Section 5. Finally, some conclusions are drawn in Section 6, and further extensions are considered.

2. DESIGN CRITERIA AND ALGORITHM

The design criteria for the new grid generation strategy may be summarized as follows:

- The geometric flexibility of the unstructured grid generator should not be compromised for Navier-Stokes meshes. This implies using unstructured grids for the surface discretization.
- The manual input required for a desired Navier-Stokes mesh should be as low as that used for the Euler case. In the present case, this requirement is solved by specifying at the points of the background grid the boundary layer thickness and the geometric progression normal to the surface.
- The generation of the semi-structured grid should be fast. Experience shows that usually more than half of the elements of a typical Navier-Stokes mesh are located in the boundary-layer regions. This requirement is met by constructing the semi-structured grids with the same normals as encountered on the surface (see Figure 1), i.e. without recurring to smoothing procedures as the semi-structured mesh is advanced into the field [10,12].
- The element size and shape should vary smoothly when going from the semi-structured to the fully unstructured mesh regions. How to accomplish this is the main topic of this paper, and is detailed in subsequent sections.
- The grid generation procedure should avoid all of the problems typically associated with the generation of Navier-Stokes meshes for regions with high surface curvature: negative or deformed elements due to converging normals, and elements that get too large due to diverging normals at the surface. In order to circumvent these problems, the same techniques which are used to achieve a smooth matching of semi-structured and unstructured mesh regions are used.

Given these design criteria, as well as the approaches used to meet them, the present grid generation algorithm can be summarized as follows (see Figure 1):

M.1 Given a surface definition and a background grid, generate a surface triangulation using an unstructured grid generator.

M.2 From the surface triangulation, obtain the surface normals.

M.3 Smooth the surface normals in several passes in order to obtain a more uniform mesh in regions with high surface curvature.

M.4 Construct a semi-structured grid with the information provided by the background grid and the smoothed normals.

M.5 Examine each element in this semi-structured region for size and shape; remove all elements that do not meet certain specified quality criteria.

M.6 Examine whether elements in this semi-structured region cross each other; if so, keep the smaller elements and remove the larger ones, until no crossing occurs.

M.7 Examine whether elements in this semi-structured region cross boundaries; if so, remove the crossing elements.

M.8 Mesh the as yet 'empty' regions of the computational domain using an unstructured grid generator in combination with the desired element size and shape.

3. ELEMENT REMOVAL CRITERIA

The critical element of the matching algorithm described above is the development of good element removal criteria. The criteria to be considered are: element size, element shape, element overlap and element crossing of boundary faces.

3.1 Element Size

The two main types of problems encountered in semi-structured grid regions that are related to element size are elements that are either too large or negative (folded). These problems originate for different reasons, and will therefore be treated separately.

3.1.1 Large Elements

As a result of surface normals diverging close to convex surfaces very large elements (as compared to the user-defined size and shape) may appear in the semi-structured mesh regions. The situation is shown diagrammatically in Figure 2. The volume of each element in the semi-structured mesh region is compared to the element volume desired by the user for the particular location in space. Any element with a volume greater than the one specified by the user is marked for deletion. In the present case, the desired distribution of element size and shape is given by a background grid. Tree-search algorithms are used to relate the information between this background grid and a particular location in space (see [13] for details).

3.1.2 Negative Elements

As a result of folding away from concave surfaces, elements with negative jacobians may appear. The situation is shown diagrammatically in Figure 3. As before, the element volumes are computed. All elements with negative volumes are marked for deletion.

We have observed that typically the elements adjacent to negative elements tend to be highly deformed. Therefore, we also remove all elements that have points in common with negative elements. Obviously, this one-pass procedure can be extended to several passes, i.e. neighbours of neighbours, etc. Our experience indicates, however, that one pass is sufficient for most cases.

3.2 Element Shape

The aim of a semi-structured mesh close to a wall is to provide elements with very small size normal to the wall and reasonable size along the wall. Due to different meshing requirements along the wall (e.g. corners, separation points, leading and trailing edges for small element size, other regions with larger element size), elements that are longer in the direction normal to the wall than along the wall may appear. The situation is shown diagrammatically in Figure 4. For the semi-structured grids, the element and point numbering can be assumed as known. Therefore, a local element analysis can be performed to determine whether the element has side-ratios that are consistent with boundary layer gridding. All elements that do not satisfy this criterion are marked for deletion.

3.3 Overlapping Elements

Overlapping elements will occur in regions close to concave surfaces with high curvature, or when the semi-structured grids of two close objects overlap. Another possible scenario is the overlap of the semi-structured grids of mixing wakes. The main criterion employed is to keep the smaller element whenever an overlap occurs. In this way, the small elements close to surfaces are always retained. Straightforward testing would result in $O(N_{el})$ operations per element, where N_{el} denotes the number of elements, leading to a total number of operations of $O(N_{el}^2)$. By using quad/octrees [13], or other suitable data structures [19], the number of elements tested can be reduced significantly, leading to a total number of operations of $O(N_{el} \log N_{el})$.

For quad/octrees, the complete testing procedure would look as follows:

- Construct a quad/octree for the points;
- Order the elements according to decreasing volume (e.g. in a heap-list [13]);
- Construct a linked list for all the elements surrounding each point;
- Loop over the elements, in descending volume, testing:
- IF the element, denoted in the sequel by IELEM, has not been marked for deletion before:
  - Obtain the minimum/maximum extent of the coordinates belonging to this element;
  - Find from the quad/octree all points falling into this search region, storing them in a list LCLOP(1:NCLOP);
  - Find all the unmarked elements with smaller volume than IELEM surrounding the points stored in LCLOP(1:NCLOP); this yields a list of close elements LCLOE(1:NCLOE);
  - Loop over the elements stored in LCLOE(1:NCLOE):
    - IF the element crosses the faces of IELEM or is inside IELEM: Mark IELEM for deletion

The reason for looping over the elements according to descending volumes is that the search region is obtained in a natural way, i.e. the extent of the element. Looping according to ascending volumes would imply guessing search regions.

As negative elements could lead to a failure of this test, the overlap test is performed after the negative elements have been identified and marked.

3.4 Elements Crossing Boundary Faces

In regions where the distance between surfaces is very small, the crossing of boundary faces by elements from the semi-structured region is likely to occur. As this test is performed after the element crossing tests are conducted, the only boundaries that need to be treated are those that have no semi-structured grid attached to it. In order to detect if overlapping occurs, we loop over the surface faces, seeing if any element crosses it. As before, straightforward testing would result in an expensive $O(N_{el} \cdot N_f)$ procedure, where N_f denotes the number of boundary faces. By using quad/octrees [13], this complexity can be reduced to $O(N_f \log N_{el})$. The face-crossing check looks essentially the same as the check for overlapping elements, and its explicit description is therefore omitted.

4. SUBDIVISION OF PRISMS INTO TETRAHEDRA

As we only desire to work with tetrahedra, the prisms formed by extruding the surface triangles along the smoothed normals must be subdivided. This subdivision must be performed in such a way that the diagonals introduced at the

rectangular faces of the prisms match across prisms. Given that a prism cannot be subdivided into tetrahedra in any arbitrary way, care has to be taken when choosing these diagonals. Figure 5 illustrates the possible diagonals as the base sides of the prism are traversed. One can see that in order to obtain a combination of diagonals that can be subdivided into tetrahedra, not all sides of the triangular base have to be up-down or down-up as one traverses the sides. This implies that the sides of the triangular base mesh have to be marked in such a way that no such combination occurs. We have implemented the following iterative procedure to arrive at valid side-combinations:

D.0 Given:

- The sides of the surface triangulation
- The sides of each surface triangle
- The triangles that surround each surface triangle

D.1 DO: loop over the surface triangles

- IF the current side-combination is not valid:
  - DO: loop over the sides of the triangle
    - IF the inversion of the side/diagonal orientation leads to an allowed side-combination in the neighbour-triangle:
      - Invert the side/diagonal orientation
      - Goto next element

    ENDIF

  ENDDO

ENDIF

ENDDO

D.2 IF any unallowed combinations are left: GOTO D.1

This procedure works well, converging in at most two passes over the surface mesh for all cases tested to date.

5. EXAMPLES

Blocked Channel with Object: The surface definition is given in Figure 6a. The semi-structured grid generated from the surface discretization, consisting of NELEM=7,784 element, is shown in Figure 6b. Observe that we have sharp convex and concave corners, smoothed normals, negative elements at concave corners, and element overlap between semi-structured grids. Application of the removal criteria reduced the number of elements to NELEM=4,638, yielding the mesh shown in Figure 6c. The final unstructured mesh, consisting of NELEM=8,397 elements is shown in Figure 6d,e. Notice the smooth transition between the semi-structured and unstructured grid regions.

Multi-Element Airfoil: The surface definition is shown in Figure 7a. The semi-structured grid generated from the surface discretization, is shown in Figure 7b and consisted of NELEM=57,252 elements. As before, the normals have been smoothed, but deformed elements appear due to surface curvature. Overlapped elements are present in the inter-airfoil gap regions. Application of the removal criteria reduced the number of elements to NELEM=50,022, yielding the mesh shown in Figure 7c. The final unstructured mesh, consisting of NELEM=61,667 elements is shown in Figure 7d.

Cylinder on a Flat Plate: The surface definition for this 3-D case is shown in Figure 8a. The surface of the semi-structured grid generated from the surface triangulation, which consisted of NELEM=234,990 elements, is shown in Figure 8b. Overlapping and negative elements are clearly present in this mesh. Application of the removal criteria reduced this number to NELEM=138,954 yielding the surface shown in Figure 8c. The removal of these elements required less than 1min of CPU time on an IBM-RISC-550 workstation. The surface of the final unstructured mesh, consisting of NELEM=213,979 elements is shown in Figure 8d. The CPU time required for the complete mesh was less than 10min on an IBM-RISC-550 workstation. This mesh was used for an incompressible laminar flow simulation. A Blasius profile was prescribed at the entrance plane, and the Reynolds-number based on the cylinder diameter was set to $Re = 100$. Figures 8e,f show some of the cross-sectional meshes, as well as the solution obtained.

Generic Missile: The surface definition for this 3-D case is shown in Figure 9a. The surface of the semi-structured grid generated from the surface triangulation, which consisted of NELEM=946,800 elements, is shown in Figure 9b. Overlapping and negative elements are clearly present in this mesh. Application of the removal criteria reduced this number to NELEM=697,638 yielding the surface shown in Figure 9c. The surface of the final unstructured mesh, consisting of NELEM=997,980 elements is shown in Figure 9d. The CPU time required for the complete mesh was about 55min on an IBM-RISC-550 workstation.

6. CONCLUSIONS AND OUTLOOK

A new gridding technique for Navier-Stokes calculations involving complex geometries has been presented. The technique is based on a combination of semi-structured and unstructured meshing techniques that accommodates the strengths of these two approaches while avoiding their respective weaknesses. The technique has the advantage of being generally applicable, yielding one single unstructured final mesh for a given computational domain. At the same time, the problems usually encountered when meshing surfaces with high curvature for Navier-Stokes calculations are avoided automatically. The technique was demonstrated on several examples that were run interactively on workstations, indicating a reasonable speed for possible use within an adaptive remeshing context. Future developments will center on better wake-gridding capabilities, automation of the procedure for adaptive remeshing, and the extension to free surfaces or flexible bodies immersed in the flowfield.

7. ACKNOWLEDGEMENTS

This work was partially funded by AFOSR under contract F49620-92-J-0058, with Dr. Leonidas Sakell as the technical monitor. The support of IBM in the form of a workstation for home use is also gratefully acknowledged.

8. REFERENCES

[1] P.L. George - *Automatic Mesh Generation*; J. Wiley & Sons (1991).

[2] W. Dekoninck and T. Barth (eds.) - *AGARD Rep.* 787, *Proc. Special Course on Unstructured Grid Methods for Advection Dominated Flows*, VKI, Belgium, May (1992), Chapter 8.

[3] K. Nakahashi - FDM-FEM Zonal Approach for Viscous Flow Computations over Multiple Bodies; *AIAA*-87-0604 (1987).

[4] K. Nakahashi and S. Obayashi - Viscous Flow Computations Using a Composite Grid; *AIAA-CP*-87-1128, 8th CFD Conf., Hawaii (1987).

[5] J.F. Thompson - A Composite Grid Generation Code for General 3D Regions - The EAGLE Code; *AIAA J.* **26**, 3, 271ff (1988).

[6] J.P. Steinbrenner, J.R. Chawner and C.L. Fouts - A Structured Approach to Interactive Multiple Block Grid Generation; *AGARD-CP-464*, 8 (1990).

[7] S. Allwright - Multiblock Topology Specification and Grid Generation for Complete Aircraft Configurations; *AGARD-CP-464*, 11 (1990).

[8] F.C. Dougherty and J. Kuan - Transonic Store Separation Using a Three-Dimensional Chimera Grid Scheme; *AIAA*-89-0637 (1989).

[9] R. Meakin and N. Suhs - Unsteady Aerodynamic Simulations of Multiple Bodies in Relative Motion; *AIAA*-89-1996 (1989).

[10] K. Nakahashi - Optimum Spacing Control of the Marching Grid Generation; *AIAA*-88-0515 (1988).

[11] R. Ramamurti and R. Löhner - Simulation of Subsonic Viscous Flows Using Unstructured Grids and a Finite Element Solver;*AIAA*-90-0702 (1990).

[12] Y. Kallinderis and S. Ward - Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations; *AIAA*-92-2721 (1992).

[13] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* **4**, 123-135 (1988).

[14] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* **8**, 1135-1149 (1988).

[15] J. Peraire, K. Morgan and J. Peiro - Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD; *AGARD-CP-464*, 18 (1990).

[16] D. Mavriplis - Euler and Navier-Stokes Computations for Two-Dimensional Geometries Using Unstructured Meshes; *ICASE Rep.* **90-3** (1990).

[17] T.J. Baker - Three-Dimensional Mesh Generation by Triangulation of Arbitrary point Sets; *AIAA-CP*-87-1124, 8th CFD Conf., Hawaii (1987).

[18] M.A. Yerry and M.S. Shepard - Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique; *Int. J. Num. Meth. Eng.* **20**, 1965-1990 (1984).

[19] J. Bonet and J. Peraire - An Alternate Digital Tree Algorithm for Geometric Searching and Intersection Problems; *Int. J. Num. Meth. Eng.* (1992).

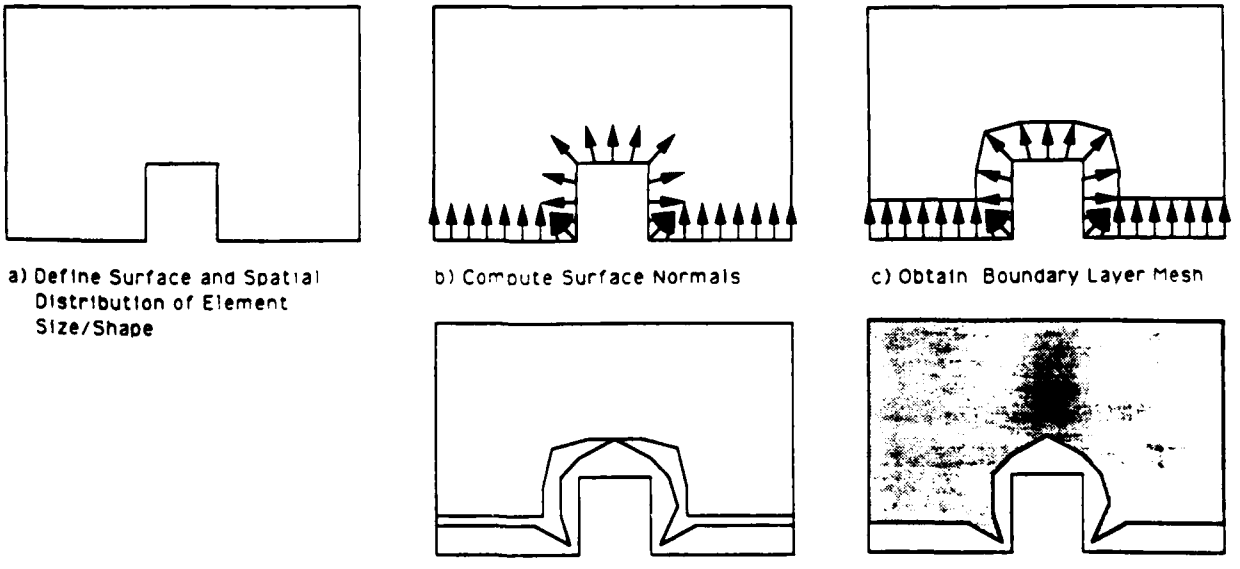


Figure 1: Grid Generation Procedure for Navier-Stokes Grids

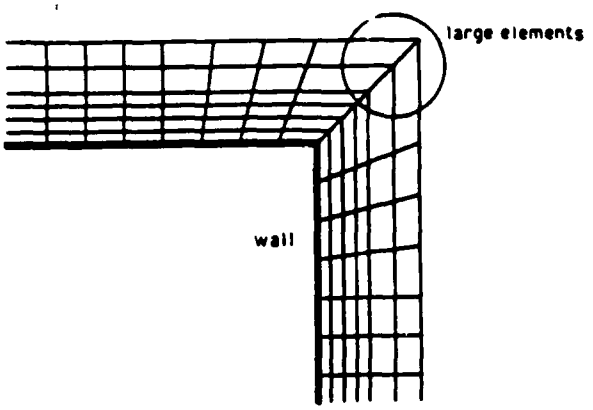


Figure 2: Removal of Large Elements

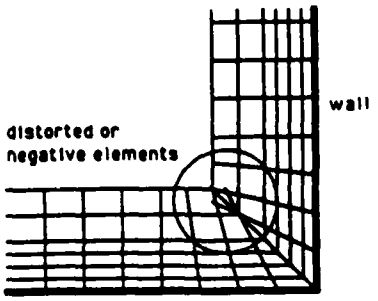


Figure 3: Removal of Negative Elements

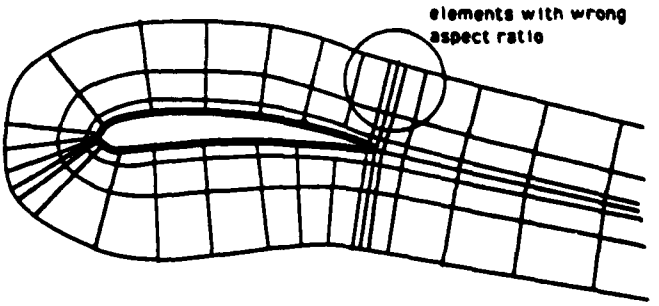
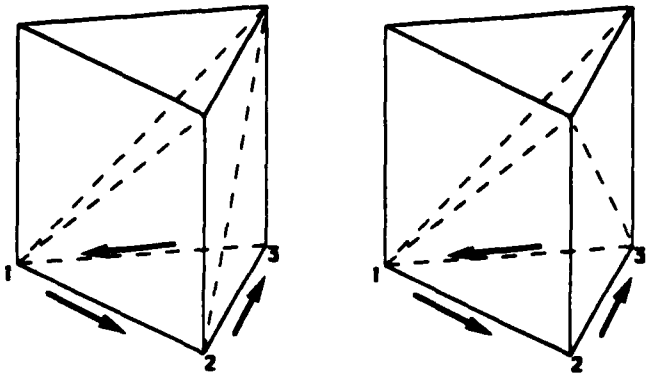


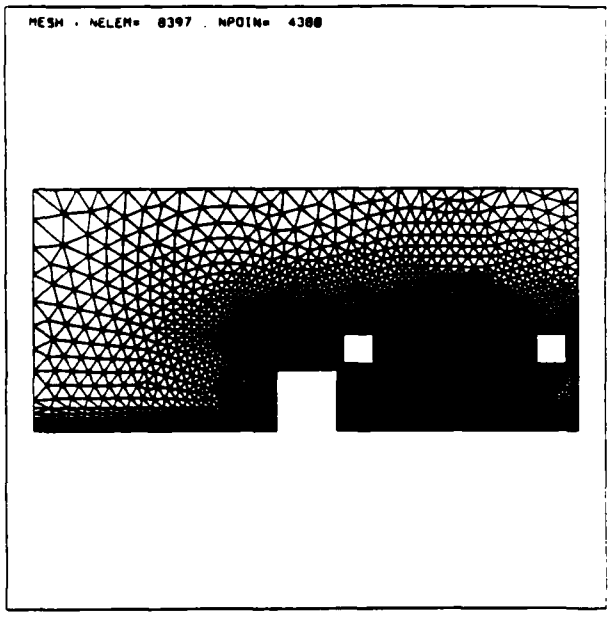
Figure 4: Removal of Elements with Wrong Aspect Ratio



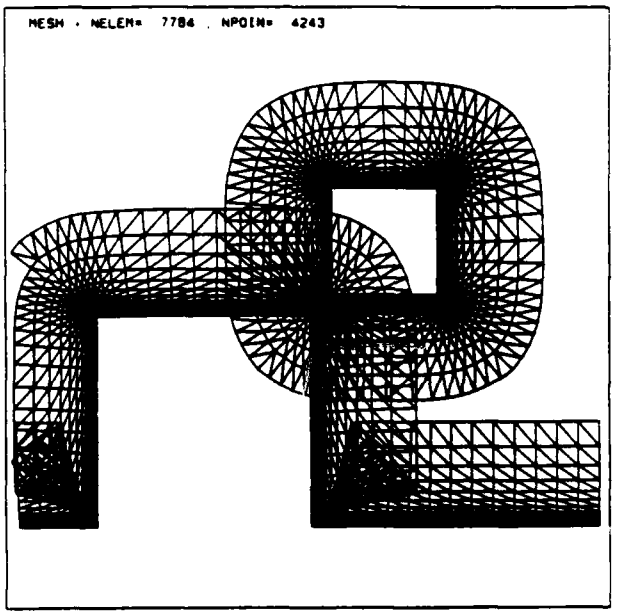
12: up
23: up
31: down

12: up
23: down
31: down

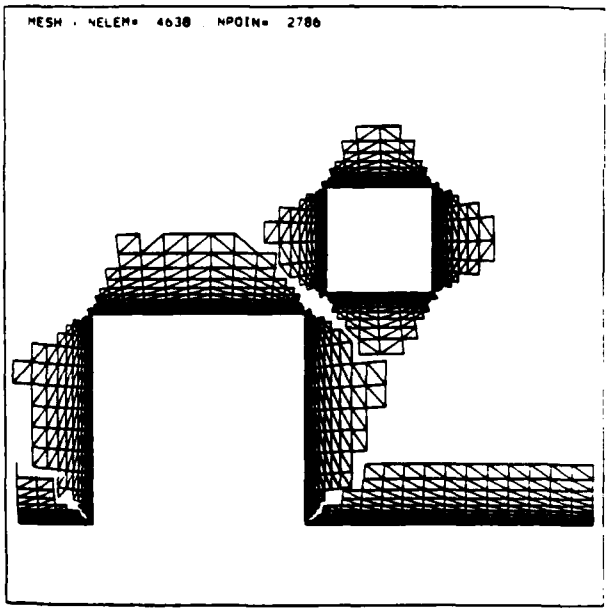
Figure 6: Generation of Tetrahedra from Prisms



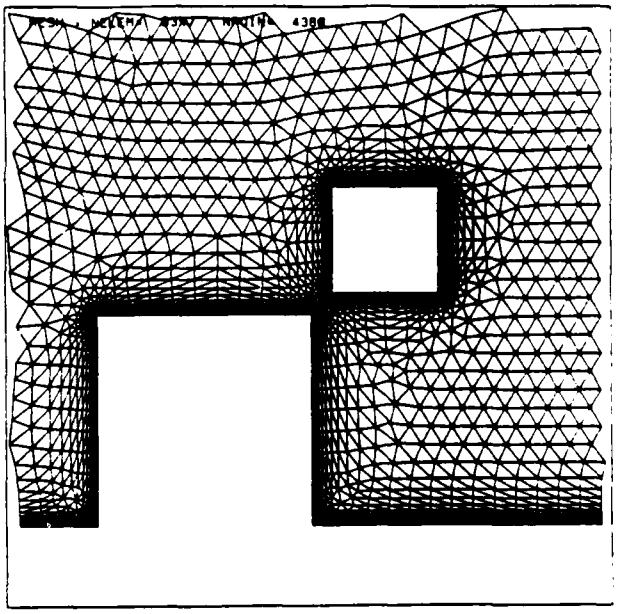
a) Surface Definition



b) Semi-Structured Mesh

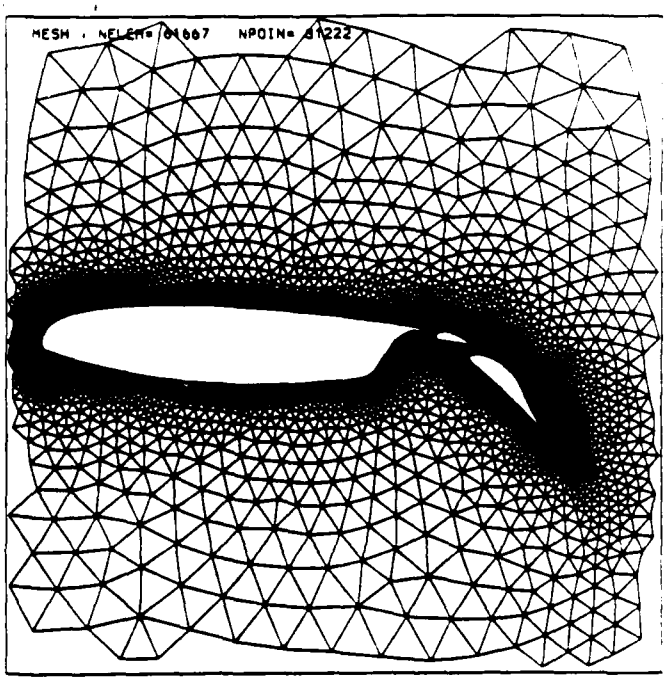


c) Mesh After Application of Element Removal Criteria

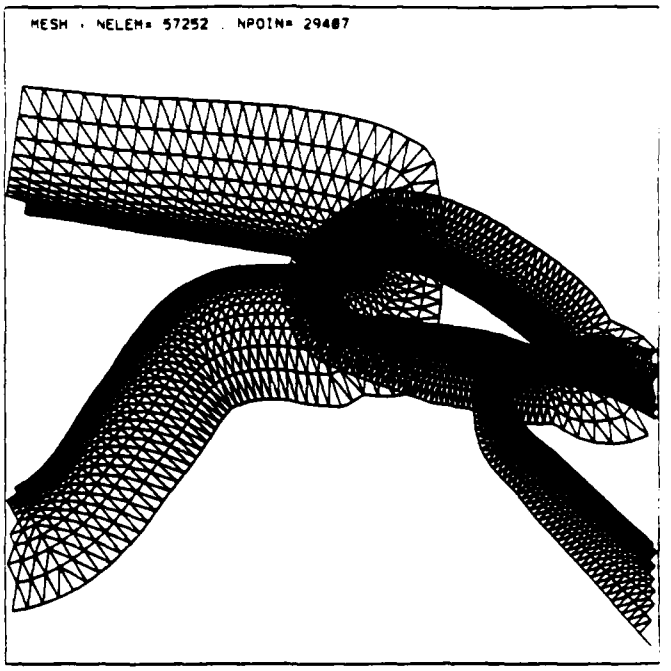


d) Final Mesh

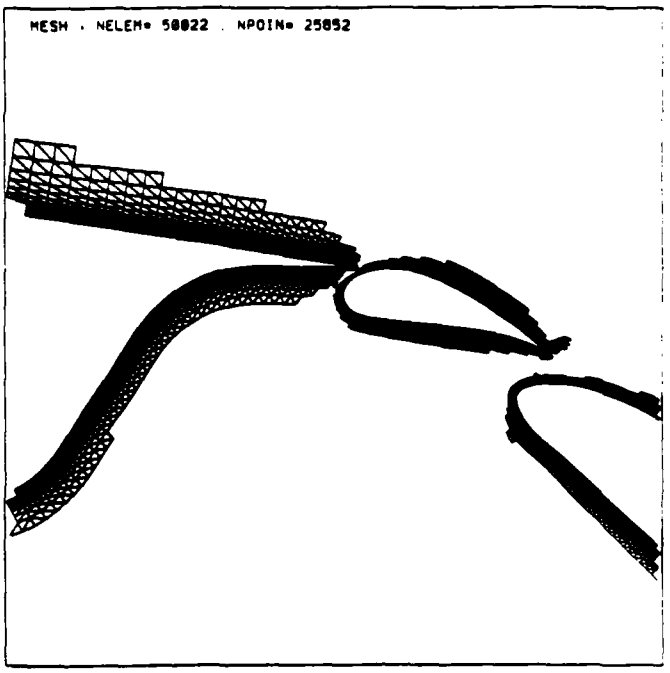
Figure 6: Blocked Channel with Objects



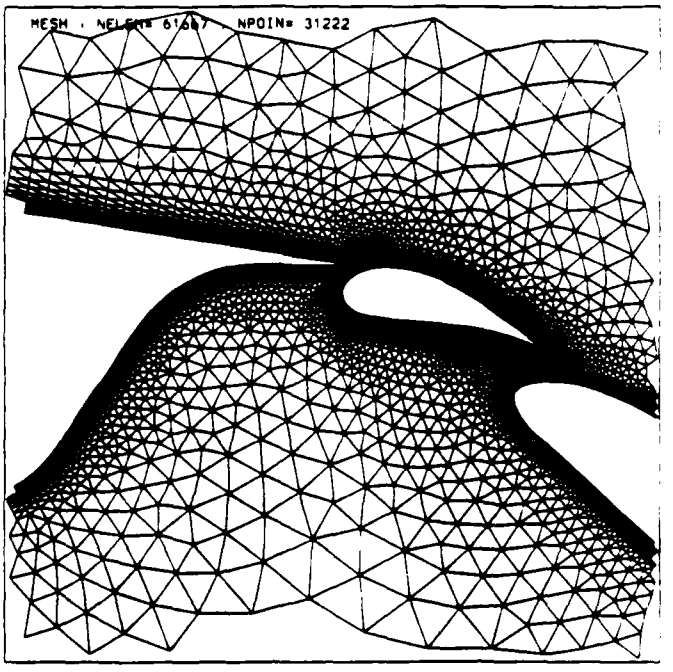
a) Surface Definition



b) Semi-Structured Mesh



c) Mesh After Application of Element Removal Criteria



d) Final Mesh

Figure 7: Multi-Element Airfoil

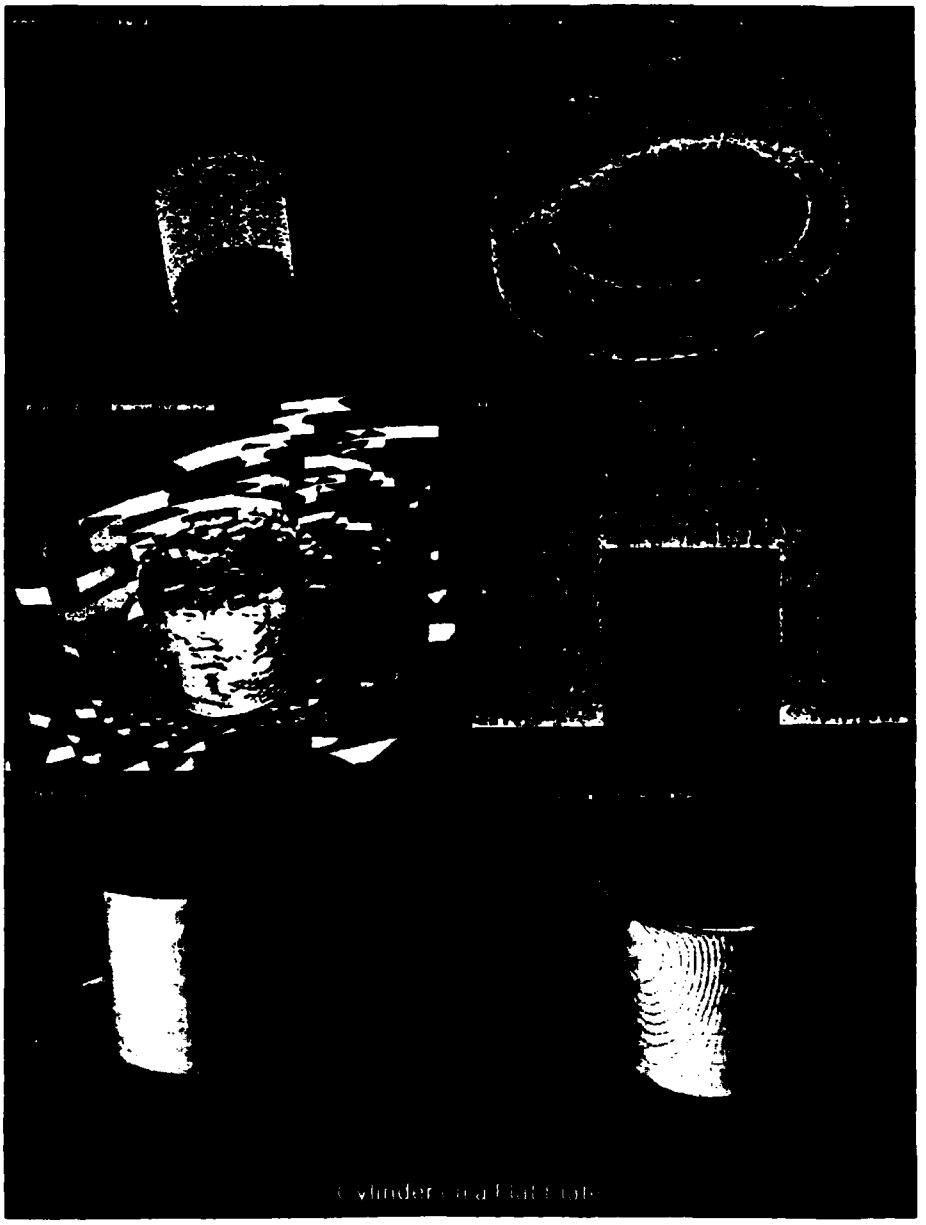


Figure 8: Cylinder on a Flat Plate

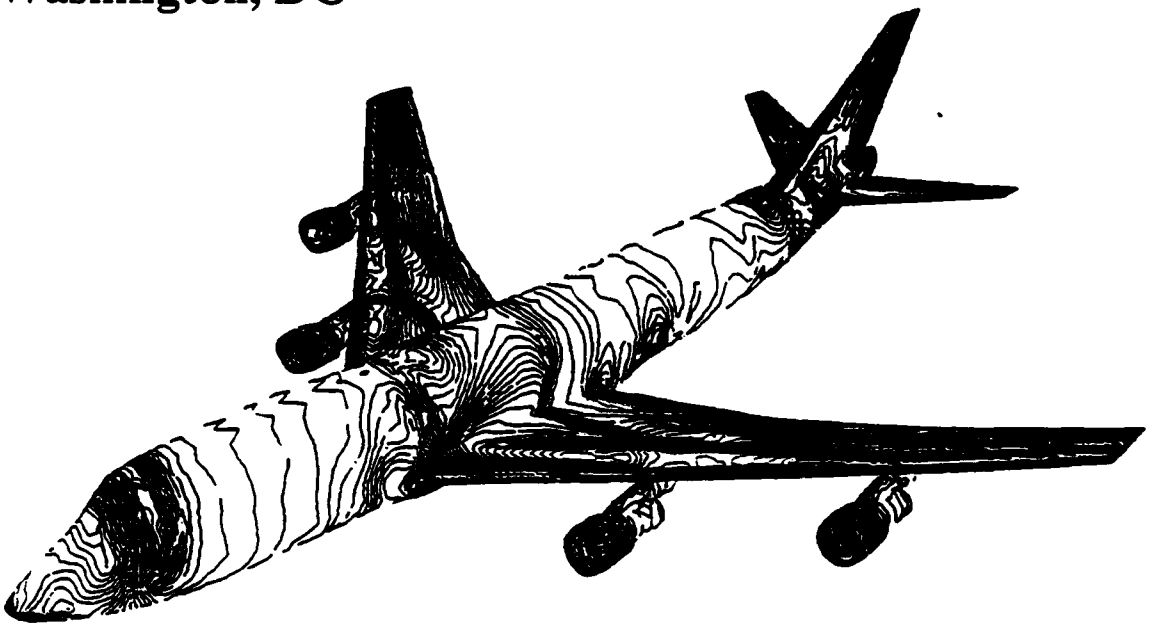
Figure 9 Generic Missile Configuration

APPENDIX 3: EDGE-BASED SOLVERS

AIAA-93-0336

Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations on Unstructured Grids

Hong Luo*, Joseph D. Baum*, Rainald Löhner†, and Jean Cabello†
*Science Applications International Corporation
McLean, VA
†George Washington University
Washington, DC



31st Aerospace Sciences Meeting & Exhibit
January 11-14, 1993 / Reno, NV

ADAPTIVE EDGE-BASED FINITE ELEMENT SCHEMES FOR THE EULER AND NAVIER-STOKES EQUATIONS ON UNSTRUCTURED GRIDS

Hong Luo and Joseph D. Baum
Science Applications International Corporation
1710 Goodridge Drive, MS 2-3-1
McLean, VA 22102, USA

and

Rainald Löhner and Jean Cabello
CMEE, School of Engineering and Applied Science
The George Washington University, Washington, D.C. 20052, USA

ABSTRACT

This paper describes recent developments of high resolution finite element schemes for the solution of the unsteady compressible Euler and Navier-Stokes equations on unstructured meshes. These finite element algorithms use an edge-based data structure, as opposed to a more traditional element-based data structure. The advantage of using such an edge-based data structure is that it provides a unified approach in which the relation between centered and upwind schemes becomes apparent, improves the efficiency of the algorithms, and reduces the storage requirements. A variety of numerical schemes using such edge-based data structure, ranging from Godunov schemes to centered schemes with blended dissipation, is presented and discussed. Adaptive mesh refinement is then added to these solvers to enhance the solution accuracy and efficiency. Various numerical results for a wide range of flow conditions, from subsonic to hypersonic in both 2D and 3D, are presented to demonstrate the performance and versatility of the proposed schemes.

1. INTRODUCTION

In recent years, significant progress has been made on developing numerical algorithms for the solution of the compressible Euler and Navier-Stokes equations. The use of unstructured meshes for computational fluid dynamics problems has become widespread due to their ability to discretize arbitrarily complex geometries and the ease with which mesh adaption can be carried out to improve the solution. However, any numerical schemes based on unstructured meshes require a storage of mesh connectivity information. This requirement leads to an increase of computer memory and the use of indirect addressing to retrieve nearest neighbor information, which, in turn, implies that any numerical algorithms will run slower on unstructured grids than on structured grids. To reduce indirect addressing, finite element schemes based on edge-based data structures have been introduced [1-4]. In addition, more sophisticated data structures such as stars, super edges, and chains have recently been developed [5]. The use of edge-based data structure has been shown to yield significant computational savings for three dimensional problems.

Extensive research has been performed during the last few years on upwind algorithms for the solution of the Euler and Navier-Stokes equations on unstructured meshes [6-9]. A significant advantage of any upwind discretization is that it is naturally dissipative, as compared with central-difference discretizations, and consequently does not require any problem-dependent parameters to adjust. So far, all the upwind schemes implemented as either node-centered or cell-centered discretizations on unstructured meshes use the finite volume approach and the control volume must be constructed first. In terms of computational efficiency, node-centered schemes are preferable to their cell-center counterparts. In the node-centered approach [6,8], the control volume is typically taken to be part of the neighboring cells that have a vertex at that node. In two dimensions, the part of the cells taken is determined by connecting the centroid of the cell and the midpoints of the two edges that share the node. In 3-D, the part of the cells taken is determined by a surface that is constructed in a similar way, a somewhat complicated geometrical process in three dimensions. The switching from element to edge-based data structure enables the implementation of upwind schemes trivial and straightforward in the context of finite elements. This is especially attractive for three dimensional problems, as there is no need to construct control volumes explicitly and geometrically.

The objective of this paper is to present recently developed high accuracy schemes on unstructured grids using an edge-based data structure. This edge-based data structure provides a unified approach in which the link between centered and upwind schemes becomes apparent. The use of such an edge-based data structure not only improves the efficiency of the algorithms, but also enables a straightforward implementation of upwind schemes in the context of finite element methods. A variety of numerical schemes using the edge-based data structure is pre-

sented and the performance of these schemes in terms of solution accuracy and overall computational efficiency is discussed. Some different strategies for the discretization of the viscous terms are considered. An approach well suited for use with an edge-based data structure is then introduced and presented. An H-refinement/coarsening adaptive scheme is implemented in these schemes to enhance the solution accuracy and efficiency. Various numerical examples for a wide range of flow conditions, from subsonic to hypersonic in both 2D and 3D, are presented to demonstrate the performance and versatility of the proposed algorithms.

2. GOVERNING EQUATIONS

The Navier-Stokes equations governing unsteady compressible viscous flows can be expressed in the conservative form as

$$\frac{\partial U}{\partial t} + \frac{\partial F^j}{\partial x_j} = \frac{\partial G^j}{\partial x_j}, \tag{2.1}$$

where the summation convention has been employed and

$$U = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, F^j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix},$$

$$G^j = \begin{pmatrix} 0 \\ \sigma_{ij} \\ u_l \sigma_{lj} + k\frac{\partial T}{\partial x_j} \end{pmatrix}. \tag{2.2}$$

Here ρ, p, e, T and k denote the density, pressure, specific total energy, temperature and thermal conductivity of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. This set of equations is completed by the addition of the equation of state

$$p = (\gamma - 1)\rho(e - \frac{1}{2}u_j u_j), \quad T = (e - \frac{1}{2}u_j u_j)/C_v\ , \tag{2.3}$$

which are valid for perfect gas, where γ is the ratio of the specific heats and C_v is the specific heat at constant volume. The components of the viscous stress tensor σ_{ij} are given by

$$\sigma_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) + \lambda\frac{\partial u_k}{\partial x_k}\delta_{ij}. \tag{2.4}$$

The thermal conductivity k and viscosity coefficient μ are assumed to be a function of the temperature and determined using Sutherland's empirical relation. It is assumed that λ and μ are related by

$$\lambda = -\frac{2\mu}{3}. \tag{2.5}$$

The left-hand side of equation (2.1) constitutes the so-called Euler equations governing unsteady compressible inviscid flows.

In the sequel, we assume that Ω is the flow domain, Γ its boundary, and n_j the unit outward normal vector to the boundary. The following boundary conditions have to be added:

On the solid wall, the slip condition is assumed for inviscid flow

$$u_i n_i = 0\ . \tag{2.6}$$

For viscous flow, no-slip condition

$$u_i = 0\ , \tag{2.7}$$

and either isothermal condition

$$T = T_0 \tag{2.8}$$

where T_0 is the total temperature or adiabatic condition

$$\frac{\partial T}{\partial x_i} n_i = 0 \tag{2.9}$$

could be imposed. In the far-field, a characteristic analysis based on the introduction of Riemann invariants for one-dimensional flow normal to the boundary is used to determine the values of the flow variables. This analysis correctly accounts for wave propagation in the far field, which is important for rapid convergence to steady-state and serves as a non-reflecting boundary condition for unsteady application.

3. VARIATIONAL FORMULATION AND FINITE ELEMENT APPROXIMATION

Let $\mathcal{T}$ be a trial function space and $\mathcal{W}$ a weighting function space, both defined to consist of all suitably smooth functions. An equivalent variational formulation of (2.1) is given by

$$\left\{ \begin{array}{l} \text{find} \quad U \in \mathcal{T} \quad \text{such that} \quad \forall W \in \mathcal{W} \\ \int_\Omega \frac{\partial U}{\partial t} W d\Omega - \int_\Omega F^j \frac{\partial W}{\partial x_j} d\Omega + \int_\Gamma F^j n_j W d\Gamma = \\ \qquad - \int_\Omega G^j \frac{\partial W}{\partial x_j} d\Omega + \int_\Gamma G^j n_j W d\Gamma\ . \end{array} \right. \tag{3.1}$$

Assuming Ω_h a classical triangulation of Ω with the nodes numbered from 1 to n and Γ_h the boundary of Ω_h, we approximate the trial and weighting spaces $\mathcal{T}$ and $\mathcal{W}$ by their subspaces of finite dimension $\mathcal{T}_h$ and $\mathcal{W}_h$, which respectively, are defined by

$$\mathcal{T}_h = \left\{ U_h(x,t) \mid U_h(x,t) = \sum_{I=1}^{n} U_I(t) N_I(x) \right\}$$

$$\mathcal{W}_h = \left\{ W_h(x) \mid W_h(x) = \sum_{I=1}^{n} \alpha_I N_I(x) \right\} \tag{3.2}$$

where N_I is the standard linear finite element shape function associated with node I, U_I is the value at

node I and α_I is a constant. The Galerkin finite element approximation is then given by

$$\begin{cases} \text{find} U_h \in \mathcal{T}_h \text{such that for each} N_I (1 \le I \le n) \\ \int_{\Omega_h} \frac{\partial U_h}{\partial t} N_I d\Omega_h = \\ \int_{\Omega_h} F^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h - \int_{\Gamma_h} F^j(U_h) n_j N_I d\Gamma_h \\ - \int_{\Omega_h} G^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h + \int_{\Gamma_h} G^j(U_h) n_j N_I d\Gamma_h \ . \end{cases} \tag{3.3}$$

The integrals appearing here are evaluated in the standard finite element form by summing individual element and boundary surface contributions, the compact support of the shape function N_I means that the equation can be written as

$$\begin{aligned} &\sum_{e \in I} \int_{\Omega_e} \frac{\partial U_h}{\partial t} N_I d\Omega_h = \\ &\sum_{e \in I} \int_{\Omega_e} F^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h - \sum_{b \in I} \int_{\Gamma_b} F^j(U_h) n_j N_I d\Gamma_h \\ &- \sum_{e \in I} \int_{\Omega_e} G^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h + \sum_{b \in I} \int_{\Gamma_b} G^j(U_h) n_j N_I d\Gamma_h, \end{aligned} \tag{3.4}$$

where the summation extends over those elements e and boundary surfaces b that contain node I. Inserting the assumed form for U_h in equation(3.4), the left-hand side integral can be evaluated exactly to give

$$\begin{aligned} \sum_{e \in I} \int_{\Omega_e} \frac{\partial U_h}{\partial t} N_I d\Omega_h &= \sum_{e \in I} [\int_{\Omega_e} N_I N_J d\Omega_h] \frac{dU_J}{dt} \\ &= M_{IJ} \frac{dU_J}{dt} \ , \end{aligned} \tag{3.5}$$

where M denotes the finite element consistent mass matrix. For steady state computations, M can be replaced by the lumped (diagonal) mass matrix, denoted by M_L.

4. EDGE-BASED FINITE ELEMENT SCHEMES FOR THE EULER EQUATIONS

It is well known that the discretization of the convective terms is crucial to the successful numerical solution of the Navier-Stokes equations. Therefore, we start by considering the solution of the Euler equations. The extension to the Navier-Stokes equations will be discussed in the next section. The integral of the convective terms appearing on the right-hand side of equation (3.4) is evaluated approximately by linearly interpolating the flux in terms of its nodal values:

$$F^j = \sum_{I=1}^{n} N_I F_I^j \ , \quad F_I^j = F^j(U_I) \ . \tag{4.1}$$

All integrals appearing in equation (3.5) can then be evaluated in closed form, leading to a matrix-vector product of the form

$$R_I = D_{IJ}^j F_J^j \ . \tag{4.2}$$

Here D_{IJ}^j are sparse matrices, whose off-diagonal entries can be identified as being associated with the edges of the mesh. Moreover, it can be shown that for any interior node, equation (3.4) can be written as

$$[M_L \frac{dU}{dt}]_I = \sum_{IJ}^{m_I} \frac{C_{IJ}^j}{2} (F_I^j + F_J^j) \tag{4.3}$$

where m_I is the number of edges connected to the node I. The coefficient C_{IJ} denotes the weight applied to the average value of the flux on the edge that connects nodes I and J, to obtain the contribution made by the edge to node I, whereas C_{JI} denotes the weight applied to the same quantity to obtain the contribution made by the edge to node J. These weights are computed as

$$C_{IJ}^j = \sum_{e \in IJ} \frac{\Omega_e}{4} \frac{\partial N_I}{\partial x_j} |_e \ , \tag{4.4}$$

where now the summation extends only over those elements Ω_e, which contain edge IJ. It can be easily verified that these weights possess the properties

$$\sum_{IJ}^{m_I} C_{IJ}^j = 0 \qquad \text{for all } I \ , \tag{4.5}$$

$$C_{IJ}^j = -C_{JI}^j \qquad \text{for all } I \text{ and } J \ . \tag{4.6}$$

For notational convenience, we define the vector C_{IJ} by the expression

$$C_{IJ} = (C_{IJ}^1, C_{IJ}^2, C_{IJ}^3) \ , \tag{4.7}$$

and let L_{IJ} denote the modulus and S_{IJ} denote a unit vector in the direction of C_{IJ}, then equation (4.3) can be written as

$$[M_L \frac{dU}{dt}]_I = \sum_{IJ}^{m_I} L_{IJ} (F_I + F_J) \tag{4.8}$$

where

$$F_I = (F_I^1, F_I^2, F_I^3).S_{IJ} \tag{4.9}$$

$$F_J = (F_J^1, F_J^2, F_J^3).S_{IJ} \ . \tag{4.10}$$

4.1 Edge-Based Data Structure

The alternative procedure for obtaining the discrete form of the equations is now apparent. While with the element-based data structure information is gathered from all the nodes of each element, operated on the element, and then scattered back to the nodes of the element, the edge-based algorithm gathers information from all the nodes of each edge, operates it on the edge, and then scattes it back to the nodes of the edge. A significant reduction in gather/scatter costs and memory requirements can

be realized by going from an element-based to an edge-based data structure. For a typical mesh of triangle elements with N nodes, the number of elements, Nelem is about 2N and the number of edges, Nedge 3N. An element-based data structure requires 2*3*Nelem, i.e., 12N gather/scatter operations, while its edge-based counterpart needs 2*2*Nedge, i.e. 12N gather/scatter operations. Whereas there appears to be little difference between an edge-based and an element based data structure in 2-D, the situation is different in 3-D. For a typical tetrahedral mesh with N nodes, the number of elements, Nelem is about 5.5N and the number of edges, Nedge 7N. An element-based data structure requires 2*4*Nelem, i.e., 44N gather/scatter operations, while its edge-based counterpart needs 2*2*Nedge, i.e., 28N. Note that a significant gather/scatter overhead reduction is achieved using an edg-based data structure in 3D, thus leading to a remarkable CPU savings.

The memory overhead for an element-based data structure calls for storing the derivatives of the shape functions and the volume of elements, and requires a 7*Nelem, i.e. 14N vector in 2-D, and a 13*Nelem, i.e. 71.5N vector in 3-D. The memory overhead for an edge-based data structure calls for storing the first order derivatives and the Laplacan and needs a 3*Nedge, i.e. 9N vector in 2-D, and a 4*Nedge, i.e. 28N vector in 3-D. Thus an edge-based data structure requires significantly less storage overhead. In fact, the edge-based data structure, defined by a list of edges with the addresses of two nodes delimiting each edge, representes the minimum amount of information required to described the unstructured mesh.

In addition, it will be shown below that this edge-based data structure is extremely useful in the process of constructing different numerical schemes, and provides a unified approach in which the relation between centered and upwind schemes is clearly apparent. It is clear that equation (4.8) is nothing but a classic Galerkin finite element scheme, which is equivalent to a central difference type scheme. By using the results of equation (4.6), this scheme allows for the appearance of chequerboarding modes and thus suffers from numerical instabilities, unless some type of numerical dissipation in the form of artificial viscosity is introduced. To construct stable schemes for the Euler equations, we have to replace the actual flux function F_{IJ} by a consistent numerical flux $\mathcal{F}_{IJ}$, and write the right-hand-side in the form

$$RHS(I) = \sum_{IJ}^{m_I} L_{IJ}\mathcal{F}_{IJ} \quad . \tag{4.11}$$

Then, by adopting different forms for this numerical flux function, we are able to construct a number of different algorithms for the Euler equations as described below.

4.2 Godunov Scheme - Exact Riemann Solver

A stable scheme can be constructed by using the exact Riemann solver [10]. This would imply replacing (4.11) by

$$RHS(I) = \sum_{IJ}^{m_I} L_{IJ}\mathcal{F}(U_{lr}^R) \tag{4.12}$$

where U_{IJ}^R denotes the local exact solution of the Riemann problem to the Euler equations and can be expressed as

$$U_{lr}^R = Rie(U_l, U_r) \tag{4.13}$$

where we have set

$$U_r = U_I, \quad U_l = U_J \quad . \tag{4.14}$$

This is the first order Godunov scheme. A scheme of higher order accuracy can be achieved by a better approximation to U_r and U_l, e.g., via reconstruction process and monotone limiting. The major disadvantage of Godunov's approach is the extensive computational work introduced through the Riemann solver.

4.3 Roe Scheme - Approximate Riemann Solver

A first simplification can be achieved by replacing the computationally costly exact Riemann solver by an approximate Riemann solver. A variety of possibilities can be defined; here we consider one of the most popular approximate Riemann solvers, namely the flux-difference splitting of Roe [11]:

$$\mathcal{F}_{IJ} = F_I + F_J - \mid A_{IJ} \mid (U_J - U_I) \; . \tag{4.15}$$

Here $\mid A_{IJ} \mid$ denotes the standard Roe matrix evaluated in the direction S_{IJ}. It can be shown that this scheme is equivalent to the first order finite volume upwind cell-vertex scheme based on a dual mesh. Many different ways exist to achieve higher order accuracy. In the present study, a scheme of higher order accuracy is obtained by using upwind-biased interpolations of the solution U via the MUSCL approach [12]. This leads to the flux function

$$\mathcal{F}_{IJ} = F_I^+ + F_J^- - \mid A(U_I^+, U_J^-) \mid (U_J^- - U_I^+) \tag{4.16}$$

where

$$F_I^+ = F(U_I^+), \qquad F_J^- = F(U_J^-) \; . \tag{4.17}$$

The upwind-biased interpolations for U_I^+ and U_J^- are defined by

$$U_I^+ = U_I + \frac{1}{4}[(1-k)\Delta_I^- + (1+k)(U_J - U_I)] \tag{4.18}$$

$$U_J^- = U_J - \frac{1}{4}[(1-k)\Delta_J^+ + (1+k)(U_J - U_i)] \tag{4.19}$$

where the forward and backward difference operators are given by

$$\Delta_I^- = U_I - U_{I-1} = 2(\nabla U)_I \cdot \mathbf{l}^{IJ} - (U_J - U_I) \quad (4.20)$$

$$\Delta_J^+ = U_{J+1} - U_J = 2(\nabla U)_J \cdot \mathbf{l}^{IJ} - (U_J - U_I) \quad (4.21)$$

where $\mathbf{l}^{IJ} = \mathbf{x}_J - \mathbf{x}_I$ is the length vector of this edge.

The parameter k, which can be chosen to control the degree of approximation, was set to $k = 1/3$ in all calculations presented here. This correponds to a third-order upwind-biased scheme [12]. With higher order spatial accuracy, spurious oscillations in the vicinity of shock waves are expected to occur. Some form of limiting is usually required to eliminate these numerical oscillations of the solution and to provide some kind of monotonicity property. The flux limiter modifies the upwind-biased interpolation U_I and U_J and the equations (4.18) and (4.19) are replaced, respectively by

$$U_I^+ = U_I + \frac{s_I}{4}[(1 - ks_I)\Delta_I^- + (1 + ks_I)(U_J - U_I)] \quad (4.22)$$

$$U_J^- = U_J - \frac{s_J}{4}[(1 - ks_J)\Delta_J^+ + (1 + ks_J)(U_J - U_I)] \quad (4.23)$$

where s is the flux limiter. Both the Van Albada limiter and Minmod limiter were employed in this study. Three options exist concerning the choice of interpolation variables: conservative variables, primitive variables, and characteristic variables. Using limiters on characteristic variables seems to give the best results, but sacrifices some computational efficiency.

4.4. Scalar Limited Dissipation

A further possible simplification can be made by replacing the Roe matrix by its spectral radius. This leads to the choice

$$\mathcal{F}_{IJ} = F_I + F_J - |\lambda_{IJ}|(U_J - U_I) \quad (4.24)$$

for the numerical flux function, where

$$|\lambda_{IJ}| = |u_{IJ}^j \cdot S_{IJ}^j| + c_{IJ} \quad (4.25)$$

and u_{IJ}^j and c_{IJ} denote edge values, computed as nodal average, of the fluid velocity and speed of sound respectively. This can be considered as a centered difference scheme plus a second order dissipation operator, leading to a first order, monotone scheme. A higher-order scheme would be obtained by a better approximation to the 'right' and 'left' states of the 'Riemann problem', which have been set to $U_r = U_I, U_l = U_J$. This would reduce the difference between U_I, U_J, decreasing in turn the dissipation. As before, limiting would provide automatic cut-off values for the possible range of U_I^l, U_J^l that lead to monotonic solutions.

4.5. Scalar Dissipation With Pressure Sensors

Another option to reduce the magnitude of $|U_I - U_J|$ is to apply a blended second- and fourth-order damping for the dissipation [1]. This is borne by the observation that even for smooth problems, central difference schemes still require fourth order damping for stability. A scheme of this type may be written as:

$$\mathcal{F}_{IJ} = F_I + F_J - |\lambda_{IJ}|\left[U_I - U_J + \frac{\beta}{2}\mathbf{l}^{IJ}(\nabla U_I + \nabla U_J)\right] \quad , \quad (4.26)$$

where $0 < \beta < 1$ denotes a pressure sensor function of the form [1]

$$\beta = 1 - \frac{p_I - p_J - 0.5\,\mathbf{l}^{IJ}(\nabla p_I + \nabla p_J)}{|p_I - p_J| + |0.5\,\mathbf{l}^{IJ}(\nabla p_I + \nabla p_J)|} \quad . \quad (4.27)$$

We have experimented with several combinations of this sensor function. The one we favor at the present time is computed in two passes over the mesh. In the first pass, the highest of the edge-based β-values given by eqnuation (4.27) is kept for each point. In a second pass, the highest value of the two points belonging to an edge is kept as the final β-value. For $\beta = 0, 1$, second- and fourth- order damping operators are obtained respectively. We remark that although this discretization of the Euler fluxes looks like a blend of second- and fourth-order dissipation, it has no adjustable parameters.

4.6. Scalar Dissipation Without Gradients

The scalar dissipation operator presented above still requires the evaluation of gradients. This can be quite costly for Euler simulations. An alternative is the simplify the combination of second- and fourth-order damping operators by writing out explicitly these operators:

$$d_2 = \lambda_{IJ}(1 - \beta)[U_I - U_J] \quad ,$$

$$d_4 = \lambda_{IJ}\beta\left[U_I - U_J + \frac{\mathbf{l}^{IJ}}{2}(\nabla U_I + \nabla U_J)\right] \quad .$$

Performing a Taylor-series expansion in the direction of the edge, we have

$$U_I - U_J + \frac{\mathbf{l}^{IJ}}{2}(\nabla U_I + \nabla U_J) \approx \frac{l^2}{4}\left[\frac{\partial^2 U}{\partial l^2}|_J - \frac{\partial^2 U}{\partial l^2}|_I\right] .$$

This suggests the following simplification, which neglects the off-diagonal terms of the tensor of second derivatives:

$$\frac{l^2}{4}\left[\frac{\partial^2 U}{\partial l^2}|_J - \frac{\partial^2 U}{\partial l^2}|_I\right] \approx \frac{l^2}{4}[\nabla^2 U_J - \nabla^2 U_I] \quad ,$$

and leads to the familiar blend second- and fourth-order damping operators

$$\mathcal{F}_{IJ} = F_I + F_J - \mid \lambda_{IJ} \mid (1-\beta)[U_I - U_J] - \lambda_{IJ}\beta\frac{l^2}{4}[\nabla^2 U_J - \nabla^2 U_I] \quad (4.28)$$

4.7 Taylor-Galerkin scheme

Due to their importance for transient calculations, it is worthwhile to consider possible edge-based Taylor-Galerkin schemes. The essential feature of any Taylor-Galerkin scheme is the combination of time and space discretizations, leading to second-order accuracy in both time and space. An edge-based two step Taylor-Galerkin scheme can be readily obtained by adopting the numerical flux

$$\mathcal{F}_{IJ} = F(U_{IJ}^{n+\frac{1}{2}}) \ , \quad (4.29)$$

where

$$U_{IJ}^{n+\frac{1}{2}} = \frac{1}{2}(U_I + U_J) - \frac{\Delta t}{2}(\frac{\partial F^j}{\partial x_j})_{IJ} \ . \quad (4.30)$$

$(\frac{\partial F^j}{\partial x_j})_{IJ}$ is computed on each edge and given by

$$(\frac{\partial F^j}{\partial x_j})_{IJ} = (F_J^j - F_I^j)l^{IJ}/l^2 \ . \quad (4.31)$$

The major advantage of this scheme is its speed, since there is no requirement of gradient computations and limiting procedures. An explicit numerical dissipation in the form of Lapidus viscosity is needed to model flows involving discontinuties. The Taylor-Galerkin scheme alone is of little use practically. However it provides a useful base scheme for the flux-corrected transport scheme presented below.

4.8 Flux-Corrected Transport Scheme

The idea behind FCT is to combine a high-order scheme with a low-order scheme in such a way that the high-order scheme is employed in smooth regions of the flow, whereas the low-order scheme is used near discontinuities in a conservative way, in an attempt to yield a monotonic solution. The implementation of an edge-based FCT scheme is exactly the same as its element-based counterpart [13]. However, the use of an edge-based data structure makes the implementation more efficient, which is especially attractive for three dimensional problems. As the high-order scheme, we employ the edge-based two step Taylor-Galerkin scheme with consistent mass matrix. The converged solution can be recast into the following form:

$$M_L \Delta U_h = R + (M_L - M_C)\Delta U^h \ . \quad (4.32)$$

The low order scheme used is simply

$$M_L \Delta U_l = R + c_d(M_C - M_L)U^n \ , \quad (4.33)$$

i.e., lumped mass-matrix plus mass diffusion. Subtracting (4.33) from (4.32) yields the antidiffusive edge contributions

$$(\Delta U^h - \Delta U^l) = M_L^{-1}(M_L - M_C)(U^n - \Delta U^h) \ . \quad (4.34)$$

This avoids any need for physical flux corrections and leads to a very fast overall scheme. Although FEM-FCT is often criticized as having a strict mathematical background, our experience has been that it gives excellent resolution for both shocks and contact discontinuities for the simulation of strongly unsteady compressible flows.

5. EXTENSION TO THE NAVIER-STOKES EQUATIONS

In this section, we extend the solution algorithms described above for the Euler equations to the solution of the Navier-Stokes equations. The numerical integration of the viscous terms is carried out in a centered way. Note that the viscous fluxes are functions of unknowns and their first derivatives, which can be expressed as

$$G^j = G^j(U, \frac{\partial U}{\partial x_k}) \ . \quad (5.1)$$

The following are three possible ways to evaluate the integrals that involve these viscous fluxes.

(a) Standard Finite Element Approach

In this approach, the integral involving the viscous fluxes is evaluated numerically after inserting the assumed form for U into the viscous flux functions. Note that over each element, $\partial U/\partial x^k$ will be constant since U is assumed to have a linear variation. Thus for an element e with nodes I, J, K and L, the integral is obtained, using a one point numerical integration rule, as

$$\int_{\Omega_e} G^j(U, \frac{\partial U}{\partial x_k})\frac{\partial N_I}{\partial x_j} d\Omega_h = \Omega_e \frac{\partial N_I}{\partial x_j}\mid_e G^j(\frac{1}{4}(U_I + U_J + U_K + U_L), \frac{\partial U}{\partial x_k}\mid_e) \ . \quad (5.2)$$

(b) Mixed Formulation

An alternative approach is to evaluate the viscous flux contributions by making using of the nodal gradients of the unknown vector U. In this case, nodal values of the viscous fluxes can be directly evaluated as

$$G^j = G^j(U, \frac{\partial U}{\partial x_k}) \ .$$

Now, the viscous fluxes can be interpolated linearly over each element and considered jointly with the inviscid fluxes. In this case, of course, the edge data structure can be readily employed to give the approximation

$$\int_{\Omega_h} G^j(U, \frac{\partial U}{\partial x_k})\frac{\partial N_I}{\partial x_j} d\Omega_h = \sum_{IJ}^{m_I} C_{IJ}^j(G_I^j + G_J^j) \ . \quad (5.3)$$

It is noted here, that the discretization of the viscous terms using the standard finite element approach only uses information from those points which are directly connected to the points being considered. On the other hand, the mixed formulation involves information from two layers of points surrounding the point under consideration. Despite their inconsistency, the numerical experience indicates that the results from two approaches are virtually identical.

(c) Edge-based Finite Element Approach.

Note that the integrals of viscous flux terms involves the evaluation of the following terms

$$\int_{\Omega_h} f \frac{\partial g}{\partial x_k} \frac{\partial N_I}{\partial x_j} d\Omega_h \ .$$

It is easily shown that this integral can be written as

$$\int_{\Omega_h} f \frac{\partial g}{\partial x_k} \frac{\partial N_I}{\partial x_j} d\Omega_h = \sum_{IJ}^{m_I} \frac{f_I + f_J}{2} \frac{D_{IJ}^j}{2} (g_I^j + g_J^j) \tag{5.5}$$

where D_{IJ} are computed as

$$D_{IJ}^j = \sum_{e \in IJ} \frac{\Omega_e}{4} \frac{\partial N_i}{\partial x_k} |_e \frac{\partial N_i}{\partial x_j} |_e \ . \tag{5.6}$$

This approach provides a way to evaluate the integral of the viscous fluxes that is consistent with the standard finite element approach. The computational effort for this approach is very small. The major disadvantage of this approach is that overhead storage requirements become significant.

6. TEMPORAL DISCRETIZATION

Equation(4.8) represents the time evalution of the unknown vector $U_I(t)$ at node I of the grid. Assuming that the nodal values U_I^n are known at time t_n, the solution is advanced over a time step Δt, to time t_{n+1} by an explicit multi-stage Runge-Kutta time-stepping scheme given by

$$U_I^{(0)} = U_I^n$$

$$\vdots$$

$$U_I^{(p)} = U_I^{(0)} - \alpha_p \Delta t (M_L)_I^{-1} R_I(U_I^{(p-1)}) \quad p = 1, 2, ..., m$$

$$\vdots$$

$$U_I^{n+1} = U_I^{(m)}$$

with the parameters α_p assigned appropriate values. The scheme is second order accurate in time. For steady state computations, implicit residual smoothing and local time-stepping are used to accelerate convergence to steady state. Residual smoothing allows the use of larger CFL numbers than the one dictated by the stability of the original scheme. For the centered-difference scheme, in the interests of computational efficiency, the diffusion contribution to the right-hand side might be evaluted only for the first stage.

7. ADAPTIVE REFINEMENT

A very attractive feature of unstructured grids is the ease with which they incorporate adaptive refinement. The addition of further degrees of freedom does not destroy any previous structure. Thus, the flow solver requires no further modification when operating on an adapted grid. For many practical problems, the regions that need to be refined are extremely small as compared to the overall domain. On the other hand, the spatial location of these regions where small elements are required are typically unknown, and may vary in time. It is thus not surprising that the use of adaptive refinement typically accrues savings in storage and CPU requirements, which range between 10-100 as compared to an overall fine mesh. Any adaptive refinement scheme consists of three different stages: determining what we want to achieve by refining the grid, developing an error indicator/estimator to detect the regions to be refined, and a refinement strategy, such as movement, enrichment or remeshing. In this study, an h-refinement scheme has been incorporated into the flow solvers in order to enhance the solution accuracy and efficiency. Further detail and description about this adaptive scheme can be found in [14-15].

8. NUMERICAL EXAMPLES

The results obtained by the centered difference schemes are not included in this paper, since such results can be found in [1] for steady state computations, and more recently in [16] for the simulation of high speed trains through tunnels using an ALE formulation and adaptive remeshing techniques. As the solutions obtained by the edge-based FEM-FCT scheme are almost identical to those obtained by its element-based counterpart, only a few test cases are presented for the edge-based FEM-FCT scheme. All the results to be presented are obtained by edge-based upwind finite element scheme, where the Van Albada limiter based on primitive variables is used for the steady state computations and the Minmod limiter based on characteristic variables is used for transient flow calculations. For the purpose of comparison , the simulations for test cases 3-4 were also perfomed using the edge-based FEM-FCT scheme to show its excellent resolution for both shock waves and contact discontinuities for the simulation of strongly unsteady compressible flows.

Test Case 1. NACA0012 Airfoil

The problem under consideration is transonic flow around a NACA0012 airfoil with a freestream Mach number of 0.85 and an angle of attack of 1 degree - a classical test problem for Euler solvers. The grid adaption scheme was used. The initial mesh containing 1,311 elements and 719 points and the final refined mesh consisting of 6,397 elements and 3,274 points after three levels of refinement are shown in

Figures 1a and 1b, respectively. The computed pressure contours in the flow fields on the initial and final adapted mesh are displayed in Figures 1c and 1d, respectively. Figures 1e and 1f show, respectively, the comparison of pressure coefficients and entropy on the airfoil between initial mesh and final refined mesh. The refinement of regions with strong gradients such as shocks, leading edge, and trailing edge is well presented and the considerable improvement in the solution for these regions after refinement is clearly apparent. The production of numerical entropy in the vicinity of the stagnation point is dramatically decreased after refinement. The pressure coefficient distribution on both initial and final adapted mesh indicates that there is only one grid point within the shock structure and demonstrates the sharp shock capturing ability of Roe's approximate Riemann solver for the solution of steady problems.

Test Case 2. Half Cylinder

In this example, we consider hypersonic flow around a half cylinder with a freestream Mach number of 10. The particular difficulty is due to the large Mach number and a quasi-rarefaction zone behind the cylinder. The final refined mesh after three levels of refinement, shown in Figure 2a, contains 20,178 elements and 10,277 points. The computed Mach number contours in the flow fields are depicted in Figure 2b. The pressure coefficient distribution and entropy on the surface are shown in Figure 2c and 2d, respectively. For this computation, the choice of Mach number as error indicator provides the best refinement in the wake of blunt bodies.

Test Case 3. Shock Tube Problem or Riemann Problem

The shock tube problem constitutes a particularly interesting and difficult test case, since it presents an exact solution to the full system of one-dimensional Euler equations containing simultaneously a shock wave, a contact discontinuity, and an expansion fan. The initial conditions in the present computation are the following:

$$\rho = 1.000, u_L = 0, p = 1.0, \quad 0.0 \leq x \leq 50.0$$
$$\rho = 0.125, u_R = 0, p = 0.1, \quad 50. < x \leq 100.$$

Figure 3 shows the computational mesh and the results for the edge-based FEM-FCT scheme and edge-based upwind finite element scheme. This is a 2D simulation of a 1D problem. The mesh consists of 101 points in the X-direction and 3 points in the Y-direction. Both schemes produced very similar results, with excellent resolution for both shock and contact discontinuity.

Test Case 4. Shock Diffraction in a Baffled Tube

The problem under consideration is shown in Figure 4. A weak shock ($M_s = 1.31$) propagates inside a baffled tube. We selected this problem to show that the FCT scheme together with the classic h-enrichement/coarsening grid adaption scheme could produce excellent results for the efficient simulation of strongly unsteady flows, without deteriorating the accuracy of the solutions. The number of refinement levels allowed in this case is 5 and the grid is modified every 7 timesteps. Density was chosen as the key variable for the error indicator. Figure 4a shows the adapted mesh at 60 μs and the experimental solution is depicted in Figure 4b. Figures 4c and 4d show the computed density contours obtained by the edge-based FEM-FCT and the edge-based upwind finite element scheme, respectively. Comparison of these results demonstrates that the edge-based FEM-FCT produced better solutions than the edge-based upwind finite element scheme, while running approximately 3 times faster.

Test Case 5. ONERA M6 Wing

In this test case, we consider a transonic flow over the ONERA M6 wing geometry. The M6 wing has a leading edge sweep angle of 30 degree, an aspect of 3.8, and a taper ratio of 0.562. The airfoil section of the wing is the ONERA "D" airfoil, which is a 10% maximum thickness-to-chord ratio conventional section. The flow solutions are presented at a Mach number of 0.84 and an angle of attack of 3.06. The mesh, which contains 179,106 grid points, 951,179 elements, and 34,013 boundary points, is depicted in Figure 5a. Figures 5b and 5c show pressure contours on the upper wing surface and lower surface, respectively. The upper surface contours clearly show the sharply captured lambda-type shock structure formed by the two inboard shock waves, which merge together near 80% semispan to form the single strong shock wave in the outboard region of the wing. The computed pressure coefficient distributions are compared with experimental data [17] at six spanwise stations in Figure 5d. The results obtained compare closely with experimental data, except at the root stations, due to lack of viscous effects.

Test Case 6. Boeing 747 Aircraft

The sixth test case is performed on a complete Boeing 747 aircraft. The 747 configuration includes the fuselage, the wing, horizontal and vertical tails, underwing pylons, and flow-through engine nacelles. The mesh, which contains 162,440 grid points, 901,275 elements and 18,454 boundary points for the half-span airplane, is shown in Figure 6a. Solutions were computed for the aircraft at a free stream of Mach number of 0.84 and an angle of attack of 2.73 degrees. The computations were performed using the three-stage Runge-Kutta time-stepping scheme with local time stepping and implicit residual smoothing. The solution was advanced 100 time-steps using a CFL number of 0.35 and 1000 time-steps with a CFL number of 4.0, to converge the solution to engineering accuracy(a decrease of a three order-of-magnitude in the L_2 norm of the density residual). The computed pressure contours on the surface of the airplane are shown in Figure 6b.

Test Case 7. Viscous Flow Past a Flat Plate

This test case involves a laminar flow past a flat plate at a Mach number of 0.5 and a chord Reynolds number of 10,000. This computation was performed to validate the Navier-Stokes code by comparing the

results with the exact Blasius solutions. The mesh used in the computation is shown in Figure 7a. It contains 2,604 elements, 1,376 points, and 146 boundary points. The computed Mach number contours in the flow field are depicted in figure 7b, where the development of a boundary layer can be clearly observed. Figure 7c shows the comparison of the Blasius velocity profile and the computed velocity profiles as scaled by the Blasius similarity law at different chord length downstream of the leading edge. The computed results indicate that the similarity solution for a flat plate boundary layer is correctly obtained and the solution agrees well with the Blasius solution. Finally, Figure 7d shows the comparison of velocity profiles obtained using two different discretization approachs for viscous terms. It is observed that these two types of discretization give practically identical results.

7. CONCLUSIONS

We have given an overview of the recent development of some high order accuracy finite element schemes to the solutions of the compressible Euler and Navier-Stokes equations on unstructured grids. These schemes are based on an edge-based data structure, as opposed to the more traditionel element-based data structure. The advantages of such data structure might be summarized as a) better performance of the numerical schemes in terms of computational efficiency and memory requirements, b) easy construction of different numerical schemes, ranging from the Godunov scheme to the centered scheme with blended dissipation, and c) easy implementation of numerical schemes for both 2D and 3D problems, due to 1-D like data structure.

ACKNOWLEDGMENTS

This research was sponsored by the Defense Nuclear Agency. Dr. Michael Giltrud served as the technical monitor. Partial funding for the last two authors was also provided by the Air Force Office of Scientific Research. Dr. Leonidas Sakell served as the technical monitor.

REFERENCES

[1] J. Peraire, J. Peiro and K. Morgan - A 3D Finite Element Multigrid Solver for the Euler Equations; *30th Aerospace Sciences Meeting*, Reno, Nevada, January 6-9, 1992.

[2] T. J. Barth - Numerical Aspects of Computing Viscous High Reynolds Number Flow on Unstructured Meshes; *29th Aerospace Sciences Meeting*, Reno, Nevada, January 7-10, 1991.

[3] V. Venkatakrishnan and D. J. Mavriplis - Implicit Solvers for Unstructured Meshes; *AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, June 24-27, 1991.

[4] J. Gaski and R.L. Collins - SINDA 1987-ANSI Revised User's Manual. Network Analysis Associates, Inc. (1987).

[5] R. Löhner - Stars, Super Edges and Chains; GWU-CMEE Report, 91/92-1, Submitted to Comp. Meth. Appl. Mech. Eng. (1992)

[6] V. Billey, J. Périaux, P. Perrier, B. Stoufflet - 2-D and 3-D Euler Computations with Finite Element Methods in Aerodynamic; *International Conference on Hypersonic Problems*, Saint-Etienne, Jan. 13-17 (1986).

[7] Timothy J. Barth and Dennis C. Jespersen - The Design and Application of Upwind Schemes on Unstructured Meshes - *AIAA-89-0366, January 9-12, 1989/ Reno, NV*

[8] David L. Whitaker, B. Grossman and R. Löhner - Two Dimensional Euler Computations on a Triangular Mesh Using an Upwind, Finite-Volume Scheme; *AIAA-89-0365, January 8-11, 1989/ Reno, NV*

[9] J. T. Batina - Three-Dimensional Flux-Split Euler Schemes Involving Unstructured Meshes; *AIAA-90-1649, January 8-11, 1990/ Reno, NV*

[10] S. Godunov, A. Zabrodine, M. Ivanov, A. Kraiko, and G. Prokopov - Résolution Numérique des Problémes Multidimensionnels de la Dynamique des Gas; USSR: Editions MIR, 1979.

[11] P. L. Roe - Approximate Riemann Solvers, Parameter Vectors and Difference Schemes; *Journal of Computational Physics*, 43 (1981), pp. 357-372.

[12] B. Van Leer - Towards the Ultimate Conservative Difference Scheme, II. Monotonicity and Conservation Combined in a Second Order Scheme; *Journal of Computational Physics*, 14 (1974), pp. 361-370.

[13] R. Löhner, K. Morgan, J. Peraire and M. Vahdati - Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations; *Int. J. Num. Meth. Fluids* 7 (1987), pp. 1093-1109.

[14] R. Löhner, K. Morgan, and O.C. Zienkiewicz - An Adaptive Finite Element Procedure for High Speed Flows; *Comp. Meth. Appl. Mech. Eng.* 51 (1985), pp. 441-465.

[15] R. Löhner - An Adaptive Finite Element Scheme for Transient Problems in CFD; *Comp. Meth. Appl. Mech. Eng.* 61 (1987), pp. 323-338.

[16] E. Mestreau, R. Löhner, and S. Aita - TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing; *31st Aerospace Sciences Meeting*, Reno, Nevada, January 11-14, 1993.

[17] V. Schmitt and F. Charpin - Pressure Distributions on the ONERA M6-Wing at Transonic Mach Numbers; in "Experimental Data Base for Computer Program Assessment," AGARD AR-138, 1979.

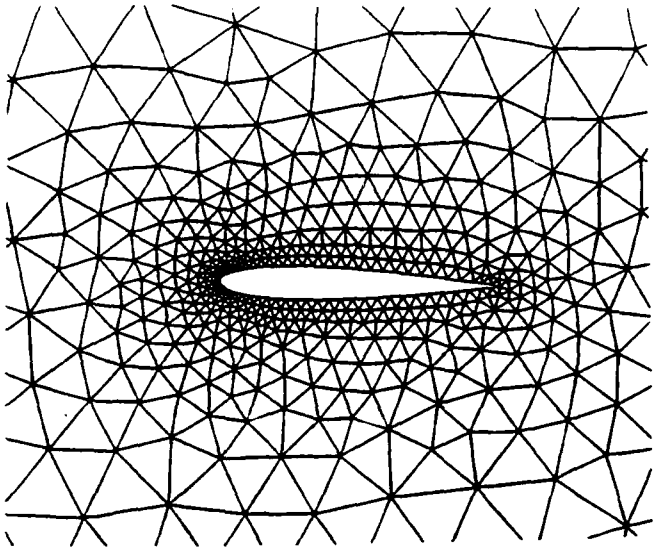


Fig.1a Initial Mesh Used for Computing Flow past NACA0012 Airfoil, nelem=1311, npoin=719

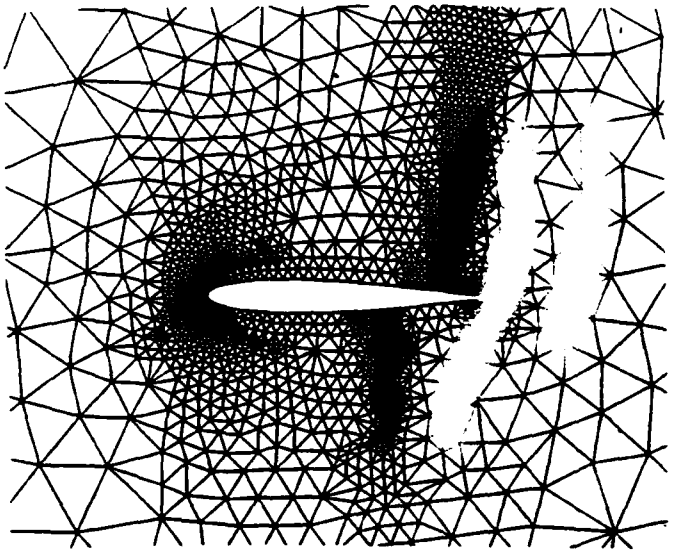


Fig.1b Final Adapted Mesh Used for Computing Flow past NACA0012 Airfoil, nelem=6397, npoin=3274

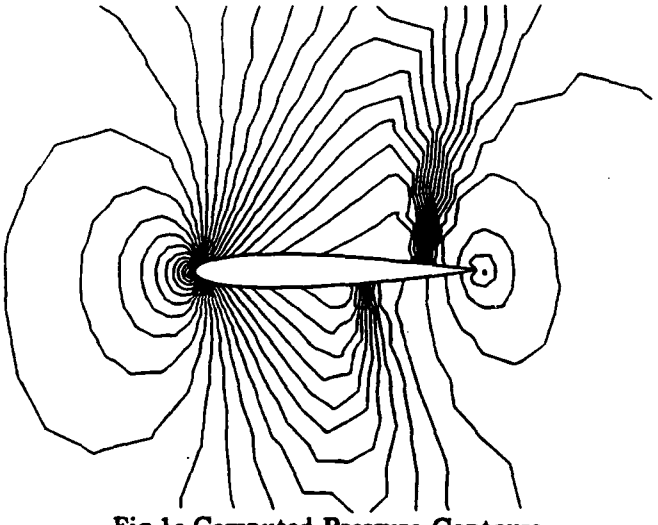


Fig.1c Computed Pressure Contours on Initial Mesh, $M_\infty = 0.85, \alpha = 1$

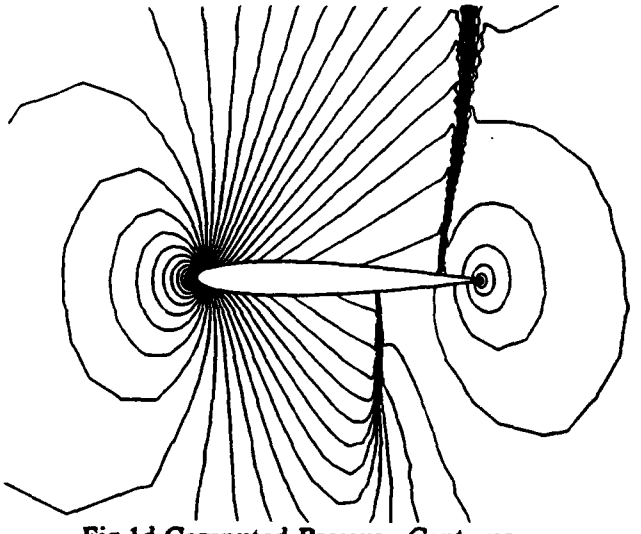


Fig.1d Computed Pressure Contours on the Final Adapted Mesh, $M_\infty = 0.85, \alpha = 1$

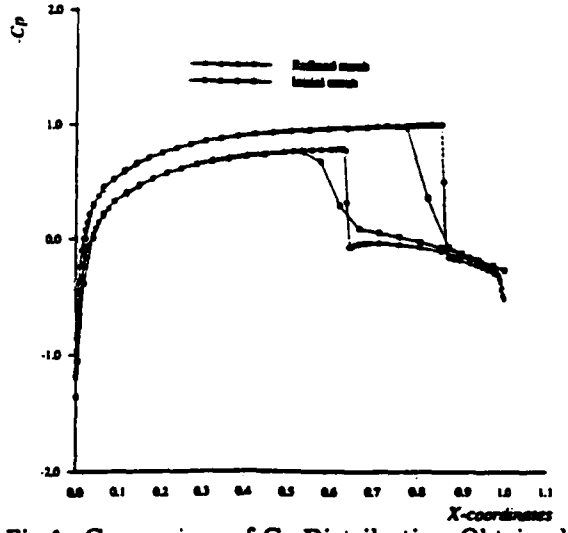


Fig.1e Comparison of C_p Distribution Obtained Using the Initial and Final Meshes

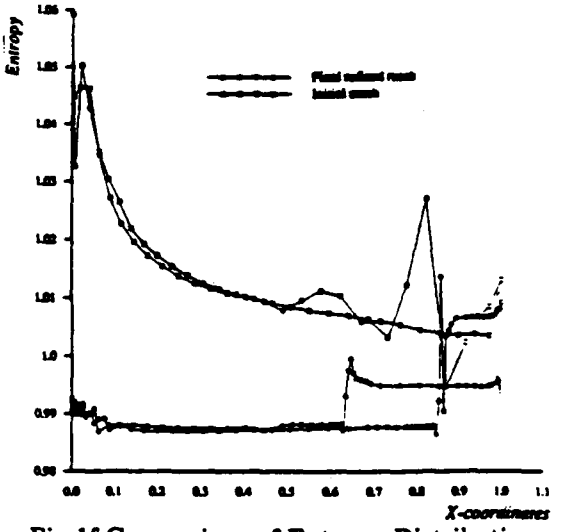


Fig.1f Comparison of Entropy Distribution Obtained Using the Initial and Final Meshes

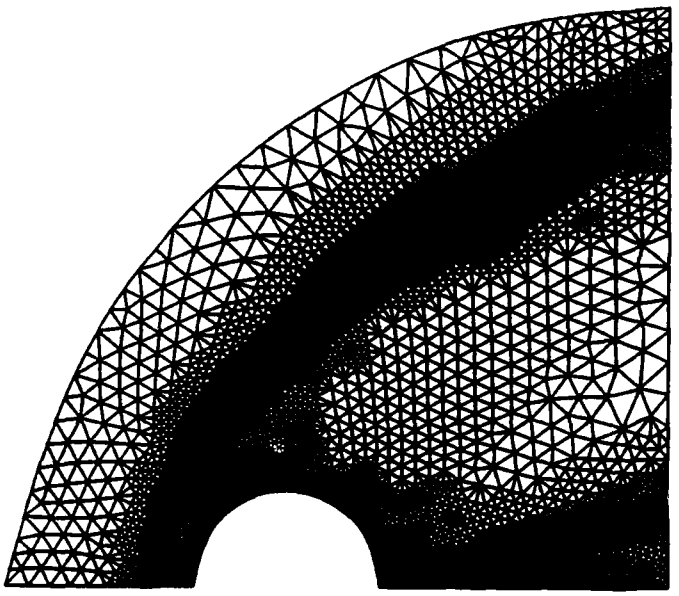


Fig.2a Final Adapted Mesh Used for Computing Flow past a Half Cylinder; nelem=20,178, npoin=10,277

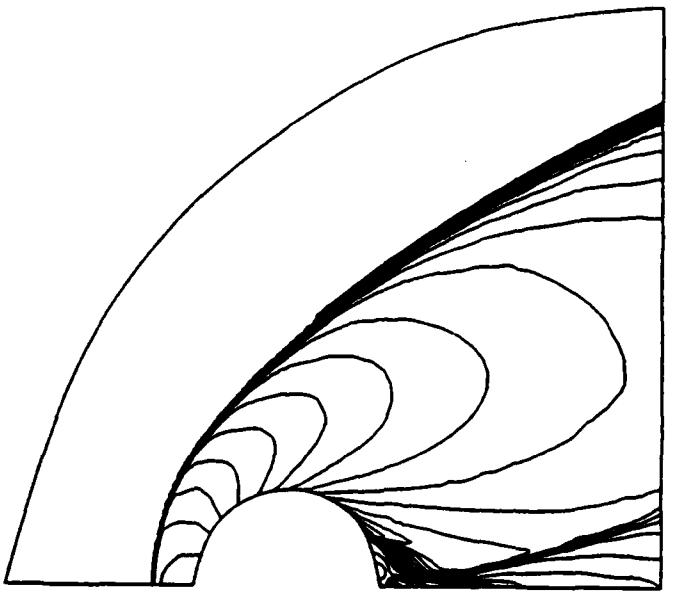


Fig.2b Computed Mach Number Contours on Final Adapted Mesh; $M_\infty = 10, \alpha = 0$

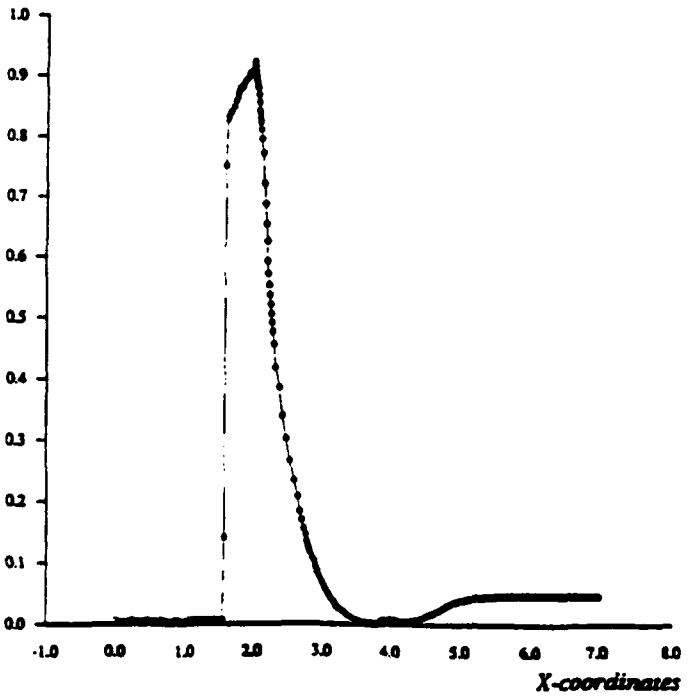


Fig.2c Computed C_p Distribution on Final Adapted Mesh; $M_\infty = 10, \alpha = 0$

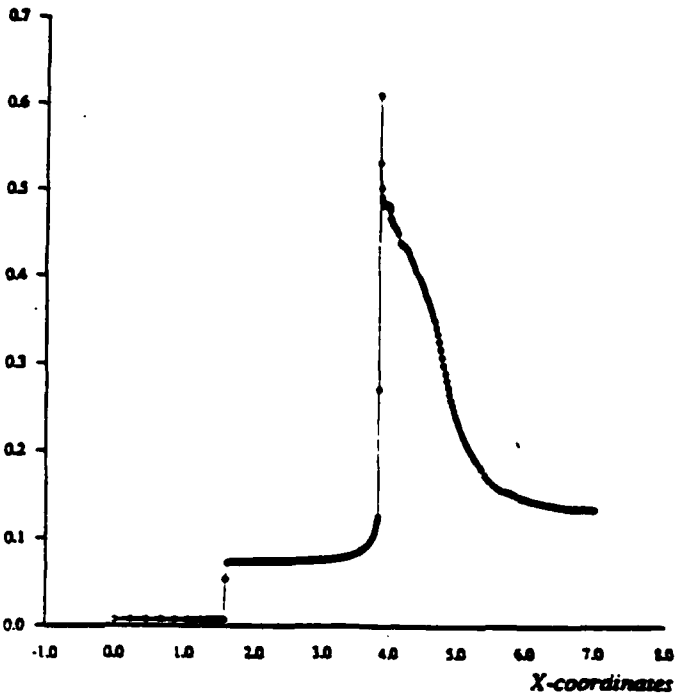


Fig.2d Computed Entropy Distribution on Final Adapted Mesh; $M_\infty = 10, \alpha = 0$

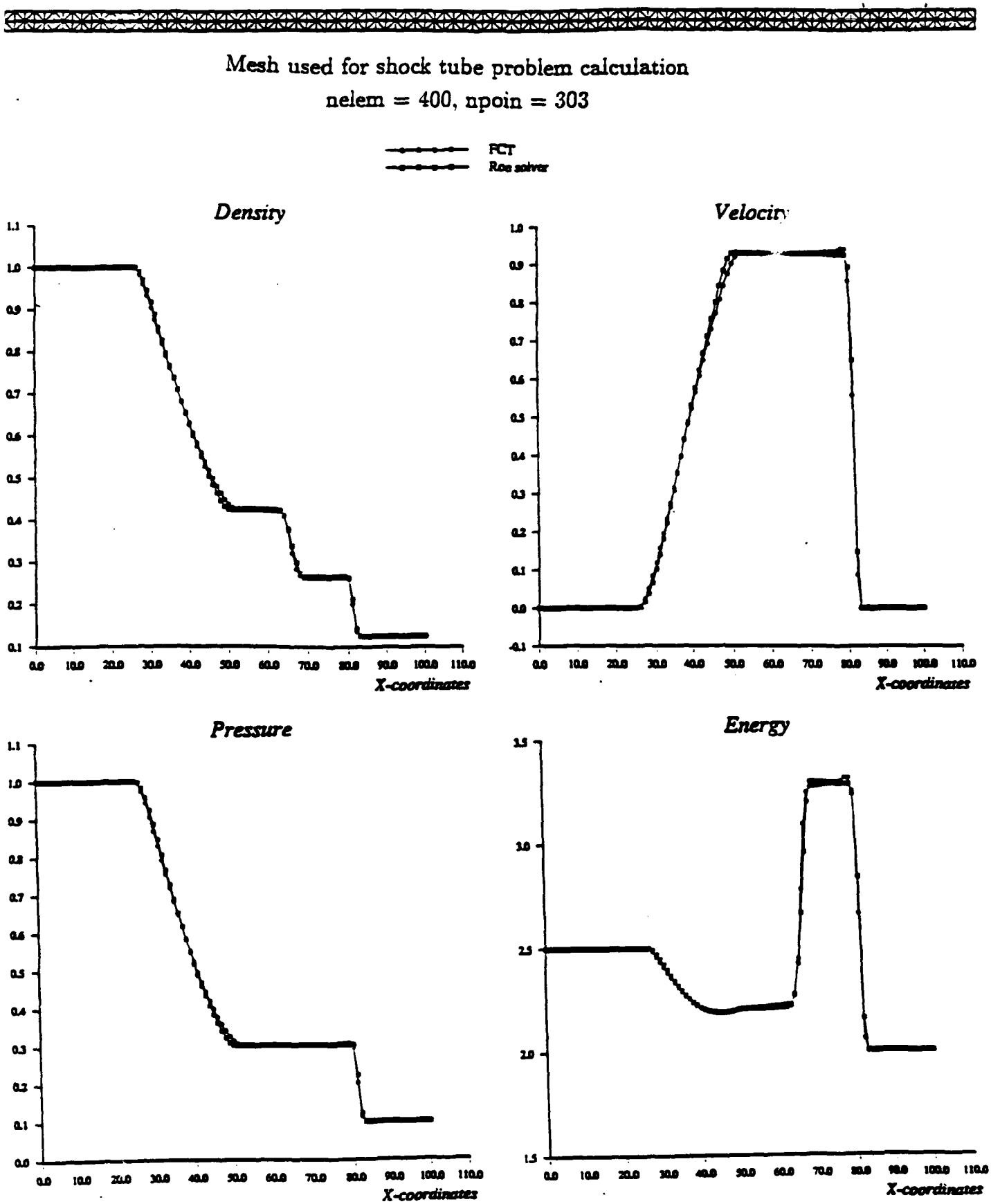


Fig. 3. Shock Tube Problem Solution Obtained Using
Edge-based FCT Scheme and Edge-based Upwind Scheme

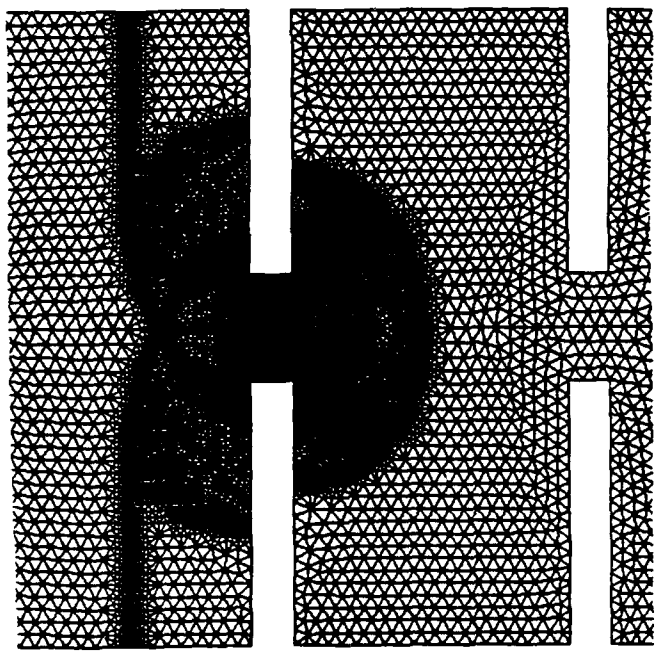


Fig.4a Adapted Mesh at 60 μs for Computing Shock diffraction in a Baffled Tube

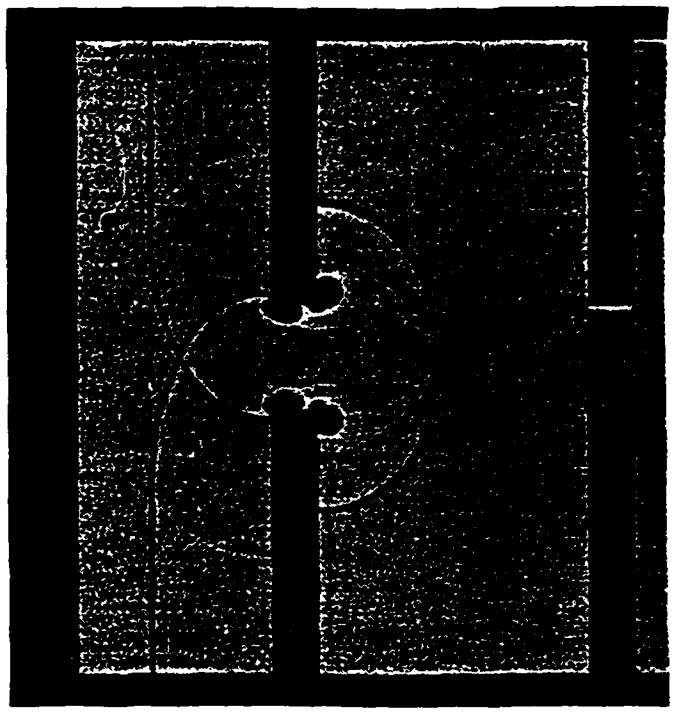


Fig.4b Shadow Cinematograph of Shock Propagation at 60 μs

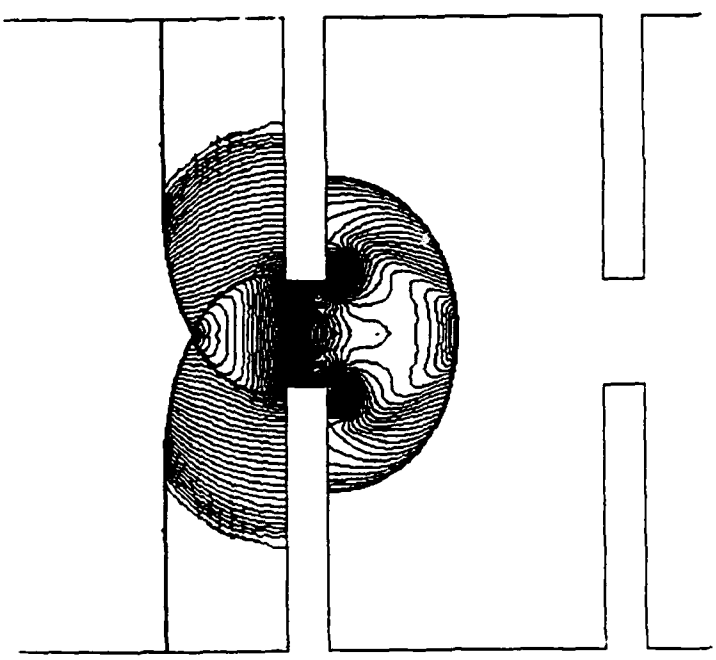


Fig.4c Computed Density Contours at 60 μs Using Edge-based Upwind Scheme

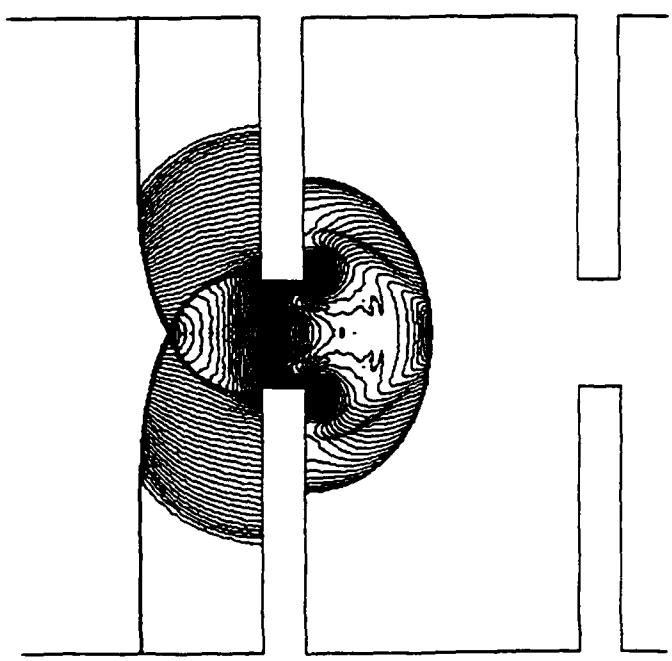


Fig.4d Computed Density Contours at 60 μs Using Edge-based FCT Scheme

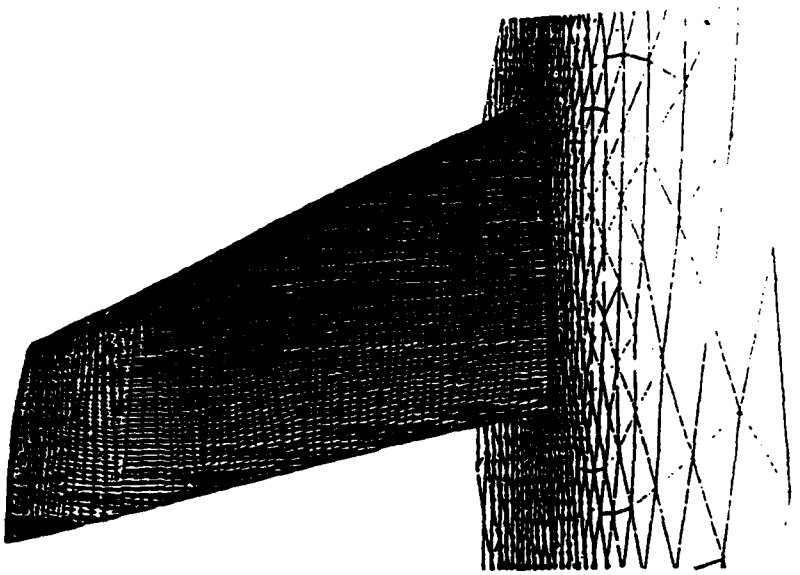


Fig.5a Mesh Used for Computing Flow past ONERA M6 Wing;
nelem=951,179, npoin=179,106, nboun=34013

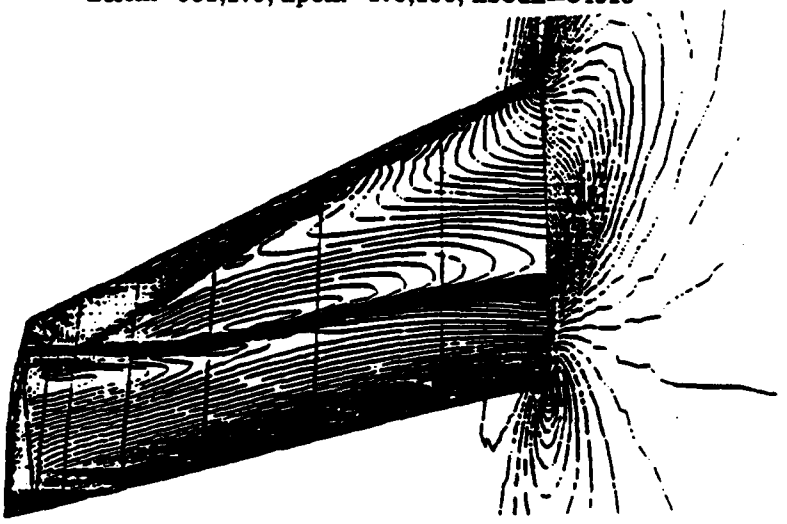


Fig.5b Computed Pressure Contours on the Upper Surface;
M_∞=0.84, α = 3.06

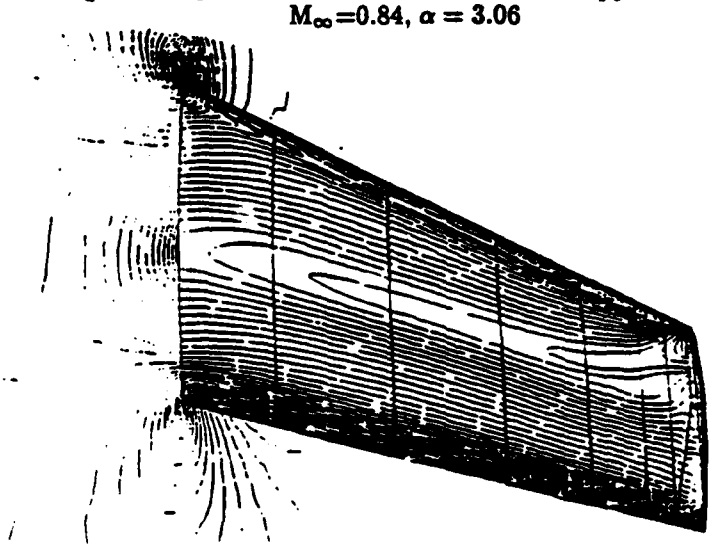


Fig.5c Computed Pressure Contours on the Lower Surface;
M_∞=0.84, α = 3.06

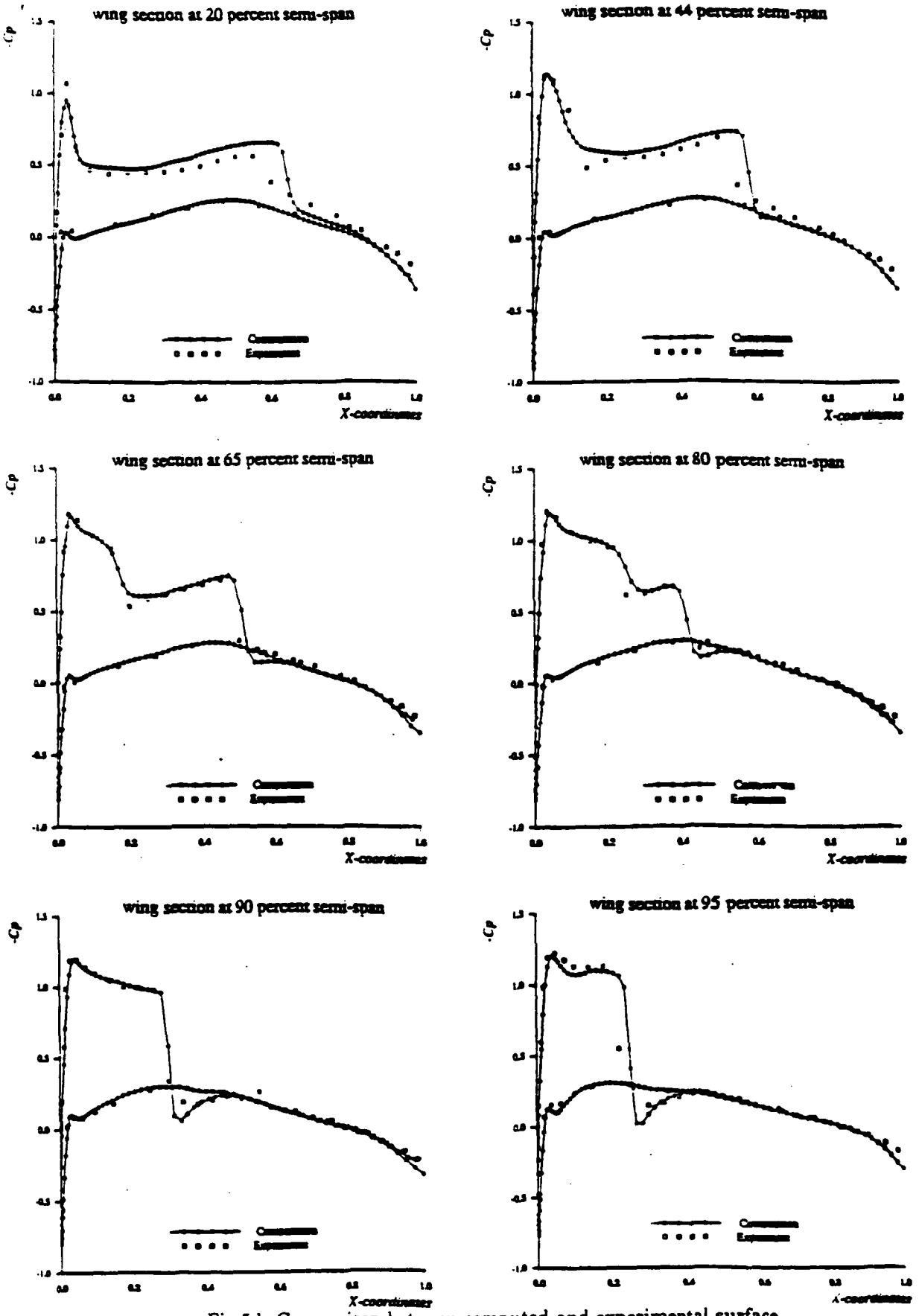


Fig.5d. Comparison between computed and experimental surface pressure coefficient for the ONERA M6 wing; $_\infty$=0.84, α = 3.06

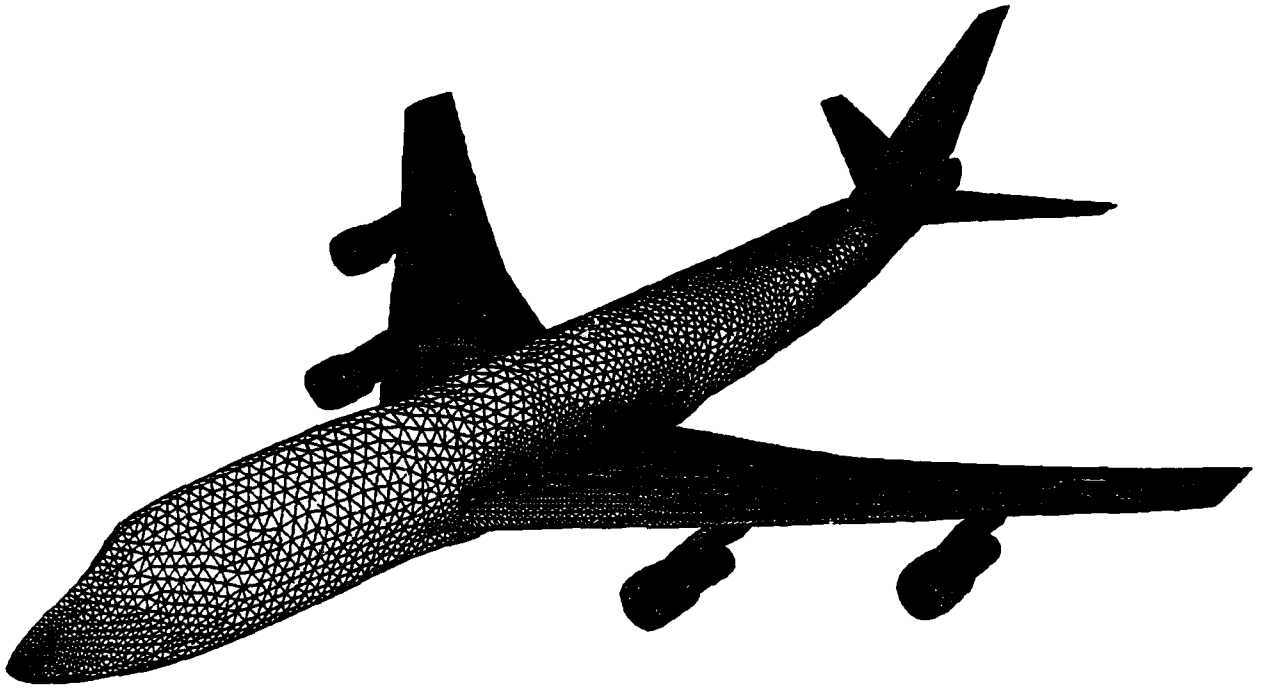


Fig.6a Surface Mesh of Triangles for the Boeing 747 Aircraft
nelem=671,382, npoin=121,612, nboun=12,025

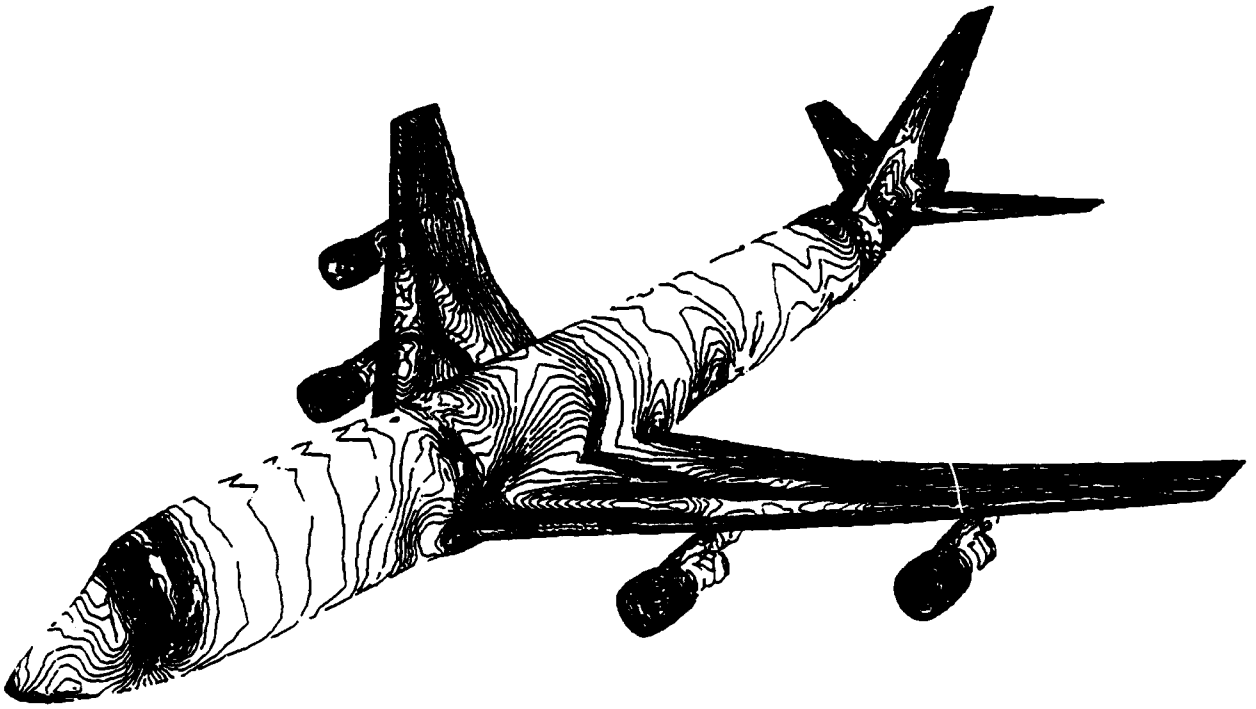


Fig.6(b) Computed Pressure Contours on the Boeing 747
at $M_\infty = 0.84, \alpha = 2.73$

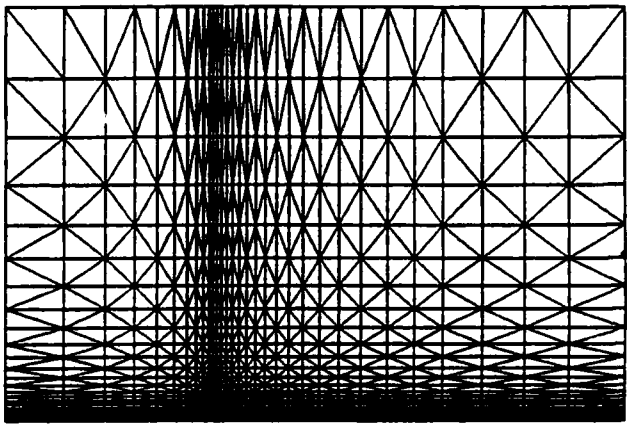


Fig.7a Mesh Used for Computing Viscous Flow past a Flat Plate; nelem=2,604, npoin=1,376

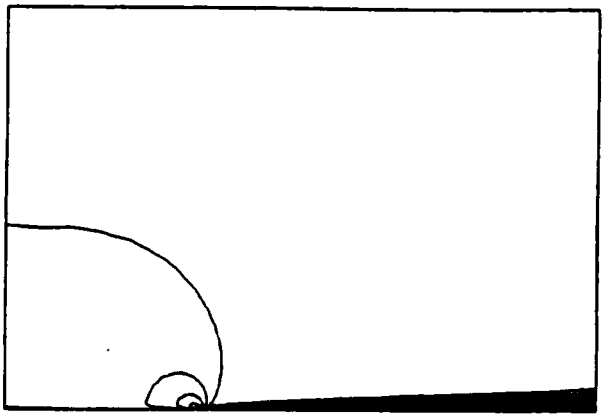


Fig.7b Computed Mach Number Contours past a Flat Plate; M_∞=0.5, Re=10,000

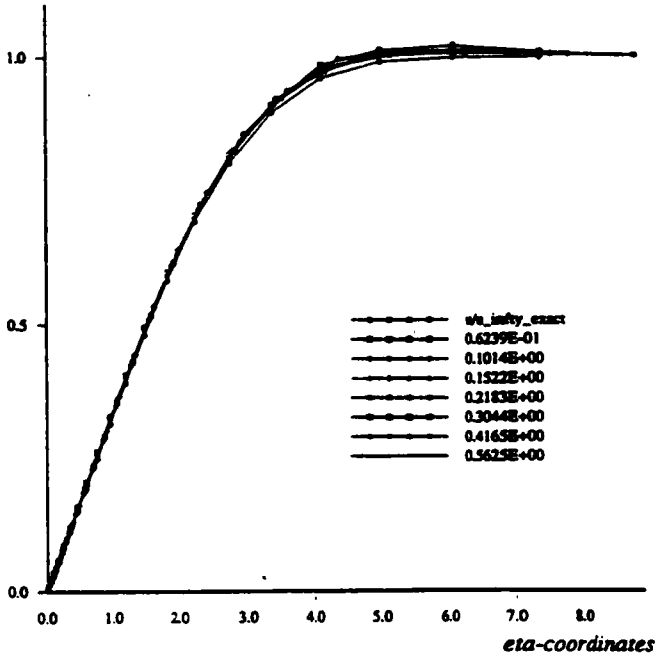


Fig.7c Streamwise Velocity Profiles past a Flat Plate; M_∞=0.5, Re=10,000

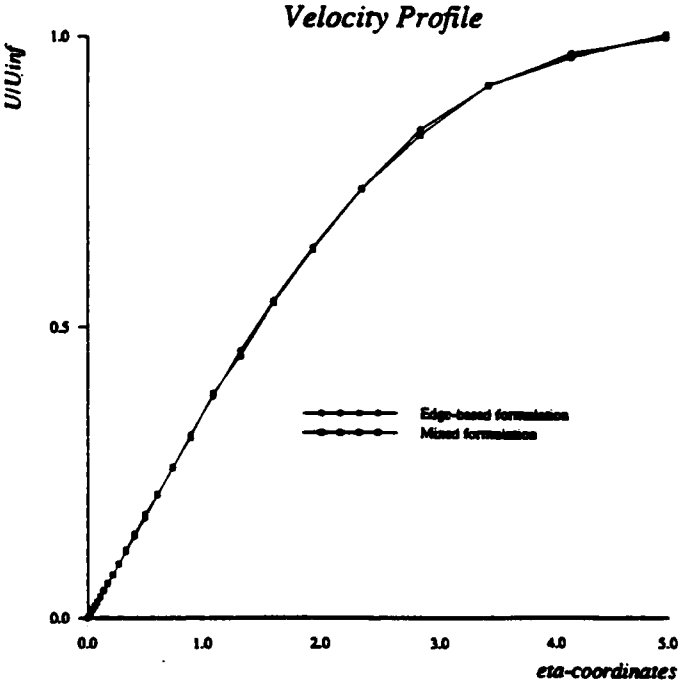


Fig.7d Comparison of Velocity Profiles Obtained Using Different Approachs to Model Viscous Terms

AIAA 93-2933

Numerical Solution of the Euler Equations for Complex Aerodynamic Configurations Using an Edge-Based Finite Element Scheme

Hong Luo*,Joseph D. Baum*, and Rainald Löhner**

* Science Applications International Corporation, McLean, VA

** George Washington University, Washington, D.C.

AIAA 24th
Fluid Dynamics Conference
July 6-9, 1993 / Orlando, FL

NUMERICAL SOLUTION OF THE EULER EQUATIONS FOR COMPLEX AERODYNAMIC CONFIGURATIONS USING AN EDGE-BASED FINITE ELEMENT SCHEME

Hong Luo and Joseph D. Baum
Science Applications International Corporation
1710 Goodridge Drive, MS 2-3-1
McLean, VA 22102, USA

and

Rainald Löhner
CMEE, School of Engineering and Applied Science
The George Washington University, Washington, D.C. 20052, USA

ABSTRACT

This paper describes the development, validation and application of a new finite element scheme for the solution of the compressible Euler equations on unstructured grids. The implementation of the numerical scheme is based on an edge-based data structure, as opposed to a more traditional element-based data structure. The use of this edge-based data structure not only improves the efficiency of the algorithm, but also enables a straightforward implementation of upwind schemes in the context of finite element methods. The algorithm has been tested and validated on some well documented configurations. A flow solution about a complete F-18 fighter is shown to demonstrate the accuracy and robustness of the proposed algorithm.

1. INTRODUCTION

In recent years, significant progress has been made in the development of numerical algorithms for the solution of the compressible Euler and Navier-Stokes equations. The use of unstructured meshes for computational fluid dynamics problems has become widespread due to their ability to discretize arbitrarily complex geometries and due to the ease of adaption in enhancing the solution accuracy and efficiency through the use of adaptive refinement techniques. However, any unstructured algorithm requires the storage of the mesh connectivity, which implies the increase of computer memory and the use of indirect addressing to retrieve nearest neighbor information. These requirements, in turn, mean that any numerical algorithm will run slower on an unstructured grid than on a structured grid. In order to reduce indirect addressing, new edge-based finite element schemes([1]-[4]) have been recently introduced. In addition, even more sophisticated data structures such as stars, super edges, and chains were recently developed by Löhner[5]. The use of edge-based data structure has shown to result in remarkable computational savings for three dimensional problems.

In the last few years, extensive research has been done on upwind type algorithms for the solution of the Euler and Navier-Stokes equations on unstructured meshes([6]-[9]). A significant advantage of upwind discretization is that it is naturally dissipative, in contrast with central-difference discretizations, and consequently does not require any problem-dependent parameters to adjust. So far, all upwind schemes implemented as either node-centered or cell-centered discretizations on unstructured meshes have used the finite volume approach where the control volume must be constructed first. In terms of computational efficiency, node-centered schemes are preferable to their cell-center counterparts. In the node-centered approach([6],[8]), the control volume is typically taken to be part of the neighboring cells that have a vertex at that node. In two dimensions, the part of the cells taken is determined by connecting the centroid of the cell and the midpoints of the two edges that share the node. In 3-D, the part of the cells taken is determined by a surface constructed in a similar way. However, this is somewhat complicated geometrically to do in three dimensions. The switching from element to edge-based data structure renders the implementation of upwind schemes trivial and straightforward in the context of the finite element approach; this is especially attractive for three dimensional application, since there is no need to construct control volumes explicitly and geometrically.

The authors have recently developed some high accuracy schemes for the solution of the Euler and Navier-Stokes equations on unstructured grids by using an edge-based data structure[1]. This paper describes the development, validation, and application of an upwind finite element algorithm to the simulation of three dimensional compressible flows around complex aerodynamic configurations. In this scheme, the spatial discretization is accomplished by an edge-based finite element formulation using Roe's flux-difference splitting. A MUSCL approach is used to achieve higher-order accuracy. A characteristic analysis based on the introduction of Riemann invariants for one-dimensional flow normal to the boundary is used to treat boundary conditions. Solutions are ad-

vanced in time by a multi-stage Runge-Kutta time-stepping scheme. Convergence is accelerated using local time-stepping and implicit residual smoothing. The algorithm has been tested and validated on some well documented configurations. A solution of the flow around a complete F-18 fighter is presented to demonstrate the accuracy and robustness of the proposed algorithm.

2. GOVERNING EQUATIONS

The Euler equations governing unsteady compressible inviscid flows can be expressed in the conservative form as

$$\frac{\partial U}{\partial t} + \frac{\partial F^j}{\partial x_j} = 0, \tag{2.1}$$

where the summation convention has been employed and

$$U = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, F^j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix}. \tag{2.2}$$

Here ρ, p, and e denote the density, pressure, and specific total energy, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. This set of equations is completed by the addition of the equation-of-state

$$p = (\gamma - 1)\rho(e - \frac{1}{2}u_j u_j), \tag{2.3}$$

which is valid for perfect gas, where γ is the ratio of the specific heats.

In the sequel, we assume that Ω is the flow domain, Γ its boundary, and n_j the unit outward normal vector to the boundary.

3. VARIATIONAL FORMULATION AND FINITE ELEMENT APPROXIMATION

Let $\mathcal{T}$ be a trial function space and $\mathcal{W}$ a weighting function space, both defined to consist of all suitably smooth functions. An equivalent variational formulation of (2.1) is given by

$$\begin{cases} \text{find} \quad U \in \mathcal{T} \quad \text{such that} \quad \forall W \in \mathcal{W} \\ \int_\Omega \frac{\partial U}{\partial t} W d\Omega - \int_\Omega F^j \frac{\partial W}{\partial x_j} d\Omega + \int_\Gamma F^j n_j W d\Gamma = 0 \ . \end{cases} \tag{3.1}$$

Assuming Ω_h is a classical triangulation of Ω with the nodes numbered from 1 to n and Γ_h the boundary of Ω_h, we approximate the trial and weighting spaces $\mathcal{T}$ and $\mathcal{W}$ by their subspaces of finite dimension $\mathcal{T}_h$ and $\mathcal{W}_h$, which respectively are defined by

$$\mathcal{T}_h = \left\{ U_h(x,t) \mid U_h(x,t) = \sum_{I=1}^{n} U_I(t) N_I(x) \right\}$$
$$\mathcal{W}_h = \left\{ W_h(x) \mid W_h(x) = \sum_{I=1}^{n} \alpha_I N_I(x) \right\} \tag{3.2}$$

where N_I is the standard linear finite element shape function associated with node I, U_I is the value at node I, and α_I is a constant. The Galerkin finite element approximation is then gi n by

$$\begin{cases} \text{find} U_h \in \mathcal{T}_h \text{such that for ea} \quad N_I (1 \le I \le n) \\ \int_{\Omega_h} \frac{\partial U_h}{\partial t} N_I d\Omega_h = \\ \int_{\Omega_h} F^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h - \int_\Gamma \quad ^j(U_h) n_j N_I d\Gamma_h. \end{cases} \tag{3.3}$$

The integrals appearing h : are evaluated in the standard finite element fc by summing individual element and boundary surface contributions, the compact support of the shape function N_I means that the equation can be written as

$$\sum_{e \in I} \int_{\Omega_e} \frac{\partial U_h}{\partial t} N_I d\Omega_h =$$
$$\sum_{e \in I} \int_{\Omega_e} F^j(U_h) \frac{\partial N_I}{\partial x_j} d\Omega_h - \sum_{b \in I} \int_{\Gamma_b} F^j(U_h) n_j N_I d\Gamma_h, \tag{3.4}$$

where the summation extends over those elements e and boundary surfaces b that contain node I. Inserting the assumed form for U_h in Eq.(3.4), the left-hand side integral can be evaluated exactly to give

$$\sum_{e \in I} \int_{\Omega_e} \frac{\partial U_h}{\partial t} N_I d\Omega_h = \sum_{e \in I} [\int_{\Omega_e} N_I N_J d\Omega_h] \frac{dU_J}{dt}$$
$$= M_{IJ} \frac{dU_J}{dt} \ , \tag{3.5}$$

where M denotes the finite element consistent mass matrix. For steady state computations, M can be replaced by the lumped (diagonal) mass matrix, denoted by M_L.

4. EDGE-BASED UPWIND FINITE ELEMENT SCHEME

It is shown in the appendix that for any interior node, Eq. (3.4) can be written as

$$[M_L \frac{dU}{dt}]_I = \sum_{IJ}^{m_I} C_{IJ}^j (F_I^j + F_J^j) \tag{4.1}$$

where m_I is the number of edges connected to the node I, and

$$C_{IJ}^j = -\sum_{e \in IJ} \frac{\Omega_e}{4} \frac{\partial N_I}{\partial x_j} |_e \ , \tag{4.2}$$

in 3D. The coefficient C_{IJ} denotes the weight applied to the average value of the flux on the edge that connects nodes I and J, to obtain the contribution made by the edge to node I, whereas C_{JI} denotes the weight

applied to the same quantity to obtain the contribution made by the edge to node J. It can be easily verified that these weights possess the properties

$$\sum_{IJ}^{m_I} C_{IJ}^j = 0 \qquad \text{for all } I\ , \tag{4.3}$$

$$C_{IJ}^j = -C_{JI}^j \qquad \text{for all } I \text{ and } J\ . \tag{4.4}$$

For notational convenience, we define the vector C_{IJ} by the expression

$$C_{IJ} = (C_{IJ}^1, C_{IJ}^2, C_{IJ}^3)\ , \tag{4.5}$$

and let L_{IJ} denote the modulus and k_{IJ} denote a unit vector in the direction of C_{IJ}, then Eq. (4.1) can be written as

$$[M_L \frac{dU}{dt}]_I = \sum_{IJ}^{m_I} L_{IJ}(F_I + F_J) = \sum_{IJ}^{m_I} L_{IJ} F_{IJ} \tag{4.6}$$

where

$$F_I = (F_I^1, F_I^2, F_I^3).k_{IJ} \tag{4.7}$$

$$F_J = (F_J^1, F_J^2, F_J^3).k_{IJ}\ . \tag{4.8}$$

The alternative procedure for obtaining the discrete form of the equations is now apparent. While with the element-based data structure information is gathered from all the nodes of each element, operated on the element, and then scattered back to the nodes of the element, the edge-based algorithm gathers information from all the nodes of each edge, operates it on the edge, and then scatters it back to the nodes of the edge. The property of conservation in the numerical scheme is guaranteed by the asymmetry of edge coefficients as expressed in Eq. (4,3). This edge-based data structure not only improves the efficiency of the algorithm [1], but also enables a straightforward implementation of upwind schemes in the context of finite element methods. It is clear that Eq. (4.6) is nothing but a classic Galerkin finite element scheme, which is equivalent to a central difference type scheme. By using the results of Eq. (4.4), this scheme allows for the appearance of chequerboarding modes and thus suffers from numerical instabilities unless some type of numerical dissipation in the form of artificial viscosity is introduced. A stable scheme can be constructed, for example using Roe's flux difference splitting [10], to replace the actual flux function F_{IJ} in Eq. (4.6) by Roe's numerical flux formula $\mathcal{F}_{IJ}$:

$$\mathcal{F}_{IJ} = F_I + F_J - \mid A_{IJ} \mid (U_J - U_I) \tag{4.9}$$

where

$$\mid A_{IJ} \mid (U_J - U_I) = \mid \Delta \bar{F}_1 \mid + \mid \Delta \bar{F}_4 \mid + \mid \Delta \bar{F}_5 \mid \tag{4.10}$$

with

$$\mid \Delta \bar{F}_1 \mid = \mid \bar{\lambda}_1 \mid \left\{ (\Delta\rho - \frac{\Delta p}{\bar{c}^2}) \begin{pmatrix} 1 \\ \bar{u} \\ \bar{v} \\ \bar{w} \\ \frac{\bar{q}^2}{2} \end{pmatrix} + \bar{\rho} \begin{pmatrix} 0 \\ \Delta u - k_x \Delta q_k \\ \Delta v - k_y \Delta q_k \\ \Delta w - k_z \Delta q_k \\ \bar{u}\Delta u + \bar{v}\Delta v + \bar{w}\Delta w - \bar{q}_k \Delta q_k \end{pmatrix} \right\} \tag{4.11}$$

$$\mid \Delta \bar{F}_{4,5} \mid = \mid \bar{\lambda}_{4,5} \mid (\frac{\Delta p \pm \bar{\rho}\bar{c}\Delta q_k}{2\bar{c}^2}) \begin{pmatrix} 1 \\ \bar{u} \pm k_x \bar{c} \\ \bar{v} \pm k_y \bar{c} \\ \bar{w} \pm k_w \bar{c} \\ \bar{h} \pm \bar{q}_k \bar{c} \end{pmatrix} \tag{4.12}$$

where $\bar{q}^2 = \bar{u}^2 + \bar{v}^2 + \bar{w}^2$, $\Delta q_k = \Delta u k_x + \Delta v k_y + \Delta w k_z$, and $\bar{q}_k = \bar{u}k_x + \bar{v}k_y + \bar{w}k_z$. The bar designates Roe-averaged quantities, which are defined by

$$\begin{aligned} \bar{\rho} &= \sqrt{\rho_I \rho_J} \\ \bar{u} &= (u_I + u_J\sqrt{\rho_J/\rho_I})/(1 + \sqrt{\rho_J/\rho_I}) \\ \bar{v} &= (v_I + v_J\sqrt{\rho_J/\rho_I})/(1 + \sqrt{\rho_J/\rho_I}) \\ \bar{w} &= (w_I + w_J\sqrt{\rho_J/\rho_I})/(1 + \sqrt{\rho_J/\rho_I}) \\ \bar{h} &= (h_I + h_J\sqrt{\rho_J/\rho_I})/(1 + \sqrt{\rho_J/\rho_I}) \\ \bar{c} &= (\gamma - 1)(\bar{h} - 0.5 * \bar{q}^2)\ . \end{aligned}$$

Furthermore, the eigenvalues of A are $\lambda_1 = \bar{q}_k$ and $\lambda_{4,5} = \bar{q}_k \pm \bar{c}$.

In order to prevent entropy violation, an entropy fix is imposed. When an eigenvalue λ reduces to zero, a smoothed value, $\mid \lambda \mid^*$, is defined to replace $\mid \lambda \mid$.

$$\mid \bar{\lambda} \mid^* = \begin{cases} \mid \bar{\lambda} \mid, & \text{if } \mid \bar{\lambda} \mid \geq \epsilon \\ \frac{\bar{\lambda}^2 + \epsilon^2}{2\epsilon}, & \text{if } \mid \bar{\lambda} \mid \leq \epsilon \end{cases} \tag{4.13}$$

where $\epsilon = K max(\lambda_J - \lambda_I, 0)$. K is a small constant.

It can be shown that this scheme is equivalent to the first order finite volume upwind cell-vertex scheme based on a dual mesh. There are many different ways to achieve higher order accuracy. In the present study, a scheme of higher order accuracy is achieved by using upwind-biased interpolations of the solution U via the MUSCL approach [11]. This leads to the flux function

$$\mathcal{F}_{IJ} = F_I^+ + F_J^- - \mid A(U_I^+, U_J^-) \mid (U_J^- - U_I^+) \tag{4.14}$$

where

$$F_I^+ = F(U_I^+), \qquad F_J^- = F(U_J^-)\ . \tag{4.15}$$

The upwind-biased interpolations for U_I^+ and U_J^- are defined by

$$U_I^+ = U_I + \frac{1}{4}[(1-k)\Delta_I^- + (1+k)(U_J - U_I)] \tag{4.16}$$

$$U_J^- = U_J - \frac{1}{4}[(1-k)\Delta_J^+ + (1+k)(U_J - U_i)] \quad (4.17)$$

where the forward and backward difference operators are given by

$$\Delta_I^- = U_I - U_{I-1} = 2(\nabla U)_I \cdot l^{IJ} - (U_J - U_I) \quad (4.18)$$

$$\Delta_J^+ = U_{J+1} - U_J = 2(\nabla U)_J \cdot l^{IJ} - (U_J - U_I) \quad (4.19)$$

where $l^{IJ} = \mathbf{x}_J - \mathbf{x}_I$ is the length vector of this edge.

The parameter k can be chosen to control a family of difference schemes in the interpolation. On structured meshes it is easy to show that $k = -1$ yields a fully upwind scheme, $k = 0$ yields semi-upwind approximation (Fromm's scheme), and $k = 1$ yields central differencing. The value $k = 1/3$ leads to a third-order-accurate upwind-biased scheme, although third-order-accuracy is strictly correct only for one-dimensional calculations. Nevertheless, $k = 1/3$ was used in the calculations presented herein. With higher order spatial accuracy, spurious oscillations in the vicinity of shock waves are expected to occur. Some form of limiting is usually required to eliminate these numerical oscillations of the solution and to provide some kind of monotonicity property. The flux limiter modifies the upwind-biased interpolation U_I and U_J and the Eqs.(4.16) and (4.17) are replaced, respectively, by

$$U_I^+ = U_I + \frac{s_I}{4}[(1-ks_I)\Delta_I^- + (1+ks_I)(U_J - U_I)] \quad (4.20)$$

$$U_J^- = U_J - \frac{s_J}{4}[(1-ks_J)\Delta_J^+ + (1+ks_J)(U_J - U_I)] \quad (4.21)$$

where s is the flux limiter. The Van Albada limiter employed in this study acts in a continuously differentiable manner and is defined by

$$s_I = max\{0, \frac{2\Delta_I^-(U_J - U_I) + \epsilon}{(\Delta_I^-)^2 + (U_J - U_I)^2 + \epsilon}\} \quad (4.22)$$

$$s_J = max\{0, \frac{2\Delta_J^+(U_J - U_I) + \epsilon}{(\Delta_J^+)^2 + (U_J - U_I)^2 + \epsilon}\} \quad (4.23)$$

where ϵ is a very small number to prevent division by zero in smooth regions of the flow. Three options exist concerning the choice of interpolation variables: conservative variables, primitive variables, and characteristic variables. Using limiters on characteristic variables seems to give the best results. However, the primitive variables are used in this study for the sake of computational efficiency.

5. BOUNDARY CONDITIONS

The treatment of boundary conditions is very important for rapid convergence to steady-state and serves as non-reflecting boundary conditions for unsteady computations.

On the solid walls, the normal velocity vanishes

$$u_n = 0 \,. \quad (5.1)$$

On the inflow and outflow, a characteristic analysis based on the 1D Riemann invariants is used to correct the computed values of the flow variables at the time step $n+1$. In the sequel, the $^{\prime}$ indicates the known linearized variables, i.e., quan ...s at the time step n, the index c designates the co: ...ted variables at the time step $n+1$, and the i... $\star$ represents the modified variables at the tim... $n+1$ after applying the boundary conditions

Note the eigenvalues and char ...eristic variables are

$$\lambda = \begin{pmatrix} v_n \\ v_n \\ v_n \\ v_n + c \\ v_n - c \end{pmatrix}, W = \begin{pmatrix} -\frac{p}{\bar{c}^2} \\ \vec{v}_t \\ (v_n + \frac{p}{\rho\bar{c}})/\sqrt{2} \\ (-v_n + \frac{p}{\rho\bar{c}})/\sqrt{2} \end{pmatrix} \quad (5.2)$$

where v_n and $\vec{v}_t = \begin{pmatrix} v_{t1} \\ v_{t2} \end{pmatrix}$ are normal and tangential velocity components. The number of boundary conditions that has to be imposed is equal to the number of negative eigenvalues. For supersonic inflow ($v_n < -c$), all the eigenvalues are negative, and therefore all the variables have to be imposed. In this case, all the variables are simply reset to freestream values. For subsonic inflow ($-c < v_n < 0$), four eigenvalues are negative, and one is positive. w_1, w_2, w_3, and w_4 are defined by the freestream values, while w_5 is determined from the computed state. The following equations are then obtained:

$$\begin{cases} \rho_\star - \frac{p_\star}{\bar{c}^2} = \rho_\infty - \frac{p_\infty}{\bar{c}^2} \\ \vec{v}_{t\star} = \vec{v}_{t\infty} \\ v_{n\star} + \frac{p_\star}{\rho\bar{c}} = v_{n\infty} + \frac{p_\infty}{\rho\bar{c}} \\ -v_{n\star} + \frac{p_\star}{\rho\bar{c}} = -v_{nc} + \frac{p_c}{\rho\bar{c}}. \end{cases} \quad (5.3)$$

By combining these equations, we get the unknown variable...

$$\begin{cases} \rho_\star = \rho_\infty + (\frac{p_\star - p_\infty}{\bar{c}^2}) \\ v_{n\star} = v_{n\infty} + (\frac{p_\infty - p_\star}{\rho\bar{c}}) \\ \vec{v}_{t\star} = \vec{v}_{t\infty} \\ p_\star = \frac{1}{2}[p_\infty + p_c + \bar{\rho}\bar{c}(v_{n\infty} - v_{nc})] \,. \end{cases} \quad (5.4)$$

For subsonic outflow ($0 < v_n < c$), only one eigenvalue is negative, and pressure is then imposed by the freestream value. w_1, w_2, w_3, and w_5 are determined from the computed values. The following relations are then obtained:

$$\begin{cases} \rho_\star = \rho_c + (\frac{p_\infty - p_c}{\bar{c}^2}) \\ v_{n\star} = v_{nc} + (\frac{p_\infty - p_c}{\rho\bar{c}}) \\ \vec{v}_{t\star} = \vec{v}_{t\infty} \\ p_\star = p_\infty \,. \end{cases} \quad (5.5)$$

For supersonic outflow ($c < v_n$), all the eigenvalues are negative, and therefore all the information comes

from the domain. In this case, nothing needs to be imposed and all the values are computed values.

6. TEMPORAL DISCRETIZATION

Equation(4.7) represents the time evaluation of the unknown vector $U_I(t)$ at node I of the grid. Assuming that the nodal values U_I^n are known at time t_n, the solution is advanced over a time step Δt, to time t_{n+1} by an explicit multi-stage Runge-Kutta time-stepping scheme given by

$$U_I^{(0)} = U_I^n$$

$$\vdots$$

$$U_I^{(p)} = U_I^{(0)} - \alpha_p \Delta t (M_L)_I^{-1} R_I(U_I^{(p-1)}) \quad p = 1, 2, ..., m$$

$$\vdots$$

$$U_I^{n+1} = U_I^{(m)}$$

with the parameters α_p assigned appropriate values. The scheme is second order accurate in time. For steady state computations, implicit residual smoothing and local time-stepping are used to accelerate convergence to steady state. The residual smoothing allows the use of larger CFL numbers than the one dictated by the stability of the original scheme. This is accomplished by averaging implicitly the residual with values at neighboring grid points. These implicit equations are solved approximately by using several Jacobi iterations. The local time-stepping uses separately a maximum allowable step size for each node according to the local stability analysis.

7. NUMERICAL RESULTS

All the grids used here were generated by the advancing front technique [12]. All computations used a three-stage Runge-Kutta time-stepping scheme with local time stepping and implicit residual smoothing. The computations were started with uniform flow and advanced with a CFL number of 4. The L_2 norm of density residual is taken as a criterion to test the convergence history.

7.1 Channel with a circular bump on the lower wall

The first test case is the well known Ni's test case. It is a transonic flow in a channel with a 10% thick circular bump on the bottom. The length of the channel is 3, its height 1, and its width 0.5. The inlet Mach number is 0.675. This is a 3D simulation of a 2D flow. This simple test case is chosen to assess both accuracy and convergence of the numerical scheme and to validate the implementation of the code. The mesh, which contains 13,891 grid points, 68,097 elements and 4,442 boundary points, is depicted in figure 1b. The convergence history is shown in Fig.1a, where a monotone convergence to computer machine zero is observed. Fig.1c displays the computed pressure contours in the flow field. The Mach number distribution on the lower wall, shown in Fig.1d indicates that there is only one grid point within the shock structure; this demonstrates the sharp shock capturing ability of Roe's approximate Riemann solver for the solution of steady problems.

7.2 Wing/pylon/finned-store configuration

The second test case is conducted for a wing/pylon/finned-store configuration reported in reference [13]. The configuration consists of a clipped delta wing with a 45 degree sweep comprised from a constant NACA64010 symmetric airfoil section. The wing has a root chord of 15 in., a semi-span of 13 in. and a taper ratio of 0.134. The pylon is located at mid-span station and has a cross-section characterized by a flat plate closed at the leading and trailing edges by a symmetrical ogive shape. The width of the pylon is 0.294 in. The four fins on the store are defined by a constant NACA0008 airfoil section with a leading edge sweep of 45 degrees and a truncated tip. The mesh used in the computation is shown in Fig.2a. It contains 274,953 grid points, 1,518,770 elements and 33,046 boundary points. The flow solutions are presented at a Mach number of 0.95 and an angle of attack of zero degree. Figures 2b and 2c show the pressure contours on the upper and lower wing surface, respectively. The computed pressure coefficient distributions are compared with experimental data at two spanwise stations in Fig.2d. The comparison with experimental data is excellent on both the upper and lower surface up to 70 percent chord. As expected from the Euler solution, the computation predicts a shock location which is downstream of that measured by the experiment due to the lack of viscous effect.

7.3 ONERA M6 Wing configuration

The third, well documented case is the transonic flow over the ONERA M6 wing configuration. The M6 wing has a leading edge sweep angle of 30 degree, an aspect of 3.8, and a taper ratio of 0.562. The airfoil section of the wing is the ONERA "D" airfoil, which is a 10% maximum thickness-to-chord ratio conventional section. The flow solutions are presented at a Mach number of 0.84 and an angle of attack of 3.06. The grid adaption scheme was used for this test case. The final adapted mesh contains 133,206 grid points, 738,669 elements, and 17,155 boundary points after two levels of refinement. The refinement of high gradient regions such as shocks, leading edge and wing tip is well captured. The final adapted upper and lower surface meshes are shown in Fig.3a. The pressure contours on the upper wing surface and lower surface, are displayed in Fig.3b, respectively. The upper surface contours clearly show the sharply captured lambda-type shock structure formed by the two inboard shock waves, which merge together near 80% semispan to form the single strong shock wave in the outboard region of the wing. The computed pressure coefficient distributions are compared with experimental data [14] at four spanwise stations in Fig.2c. The results obtained compare closely with experimental data, except at the root stations, due to lack of viscous effects.

7.4 F-18 fighter configuration

The final case is a complete F-18 fighter con-

figuration, which includes the wing, horizontal and vertical tails, and flow-through engine ducts. The mesh, which contains 93,642 grid points, 505,087 elements and 15,421 boundary points for the half-span airplane, is shown in Fig.4b. The computations were performed at a free stream of Mach number of 0.9 and an angle of attack of 3 degrees. The convergence history is depicted in figure 4a. The solution is converged to engineering accuracy (a decrease of a four order-of-magnitude in the L_2 norm of the density residual) in 1300 time-steps. It required a total of 10 CPU hours on a single processor Cray 2. The computed pressure contours on the surface of the airplane are shown in Fig.3c. Some of the features occurring in this flow regime, such as the canopy and wing shocks, are well captured.

8. CONCLUSIONS

An edge-based upwind finite element scheme has been developed for the solutions of the compressible Euler equations on unstructured grids. The numerical scheme has been tested and validated on some well documented configurations. An example application is presented for a complete F-18 fighter configuration to demonstrate the accuracy and robustness of the proposed algorithm.

ACKNOWLEDGMENTS

This research was sponsored by the Defense Nuclear Agency. Dr. Michael E. Giltrud served as the technical program monitor. Partial funding for the third author was also provided by the Air Force Office of Scientific Research. Dr. Leonidas Sakell served as the technical monitor.

REFERENCES

[1] H. Luo, J. D. Baum, R. Löhner and J. Cabello, "Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations on Unstructured meshes," *31st Aerospace Sciences Meeting*, Reno, Nevada, January 11-14,1993.

[2] J. Peraire, J. Peiro and K. Morgan, "A 3D Finite element Multigrid Solver for the Euler Equations," *30th Aerospace Sciences Meeting*, Reno, Nevada, January 6-9, 1992.

[3] T. J. Barth, "Numerical Aspects of Computing Viscous High Reynolds Number Flow on Unstructured Meshes," *29th Aerospace Sciences Meeting*, Reno, Nevada, January 7-10, 1991.

[4] V. Venkatakrishnan and D. J. Mavriplis, "Implicit Solvers for Unstructured meshes," *AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, June 24-27, 1991.

[5] R. Löhner, "Stars, Super edges and Chains," GWU-CMEE Report, 91/92-1, Submitted to Comp. Meth. Appl. Mech. Eng. (1992)

[6] V. Billey, J. Périaux, P. Perrier, B. Stoufflet, "2-D and 3-D Euler Computations with Finite Element Methods in Aerodynamic," *International Conference on Hypersonic Problems*, Saint-Etienne, Jan. 13-17 (1986).

[7] Timothy J. Barth and Dennis C. Jespersen, "The Design and Application of Upwind Schemes on Unstructured Meshes," *AIAA-89-0366, January 9-12, 1989/ Reno, NV*

[8] David L. Whitaker, "Solution Algorithms for the Two-Dimensional Euler Equations on Unstructured Meshes," *AIAA-90-0697, January 8-11, 1990/ Reno, NV*

[9] J. T. Batina, "Three-Dimensional Flux-Split Euler Schemes Involving Unstructured Meshes," *AIAA-90-1649, January 8-11, 1990/ Reno, NV*

[10] P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, 43 (1981), pp. 357-372.

[11] B. Van Leer, "Towards the Ultimate Conservative Difference Scheme, II. Monotonicity and Conservation Combined in a Second Order Scheme," *Journal of Computational Physics*, 14 (1974), pp. 361-370.

[12] R. Löhner and P. Parikh, "Generation of Three-Dimensional Unstructured Grids by the Advancing Front Method," AIAA Paper 88-0515, Jan. 1988.

[13] E. R. Heim, "CFD Wing/Pylon/Finned Store Mutual Interference Wind Tunnel Experiment," AEDC-TSR-91-P4, January 1991.

[14] V. Schmitt and F. Charpin, "Pressure Distributions on the ONERA M6-Wing at Transonic Mach Numbers," in "Experimental Data Base for Computer Program Assessment," AGARD AR-138, 1979.

APPENDIX

In this appendix, the evaluation of inviscid fluxes using an edge-based data structure is derived.

$$RHS(I) = -\int_{\Omega_h} F^j \frac{\partial N_I}{\partial x_j} d\Omega_h + \int_{\Gamma_h} F^j n_j N_I d\Gamma_h$$
$$- \int_{\Omega_h} (nnode-2) F_I^j \frac{\partial N_I}{\partial x_j} d\Omega_h$$
$$+ \int_{\Omega_h} (nnode-2) F_I^j N_I \frac{\partial}{\partial x_j} d\Omega_h, \quad (1)$$

where $nnode$ is the number of nodes in an element, and the last two integrals are artificially added in order to derive the desired formula. By using the Green's formula, we obtain

$$\int_{\Omega_h} F_I^j N_I \frac{\partial N_I}{\partial x_j} d\Omega_h = \frac{1}{2} \int_{\Gamma_h} F_I^j n_j N_I N_I d\Gamma_h. \quad (2)$$

Using Eq. (2), Eq (1) can be written as

$$\begin{aligned} & RHS(I) \\ &= -\int_{\Omega_h} (F^j + (nnode-2)F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h \\ &\quad + \int_{\Gamma_h} F^j n_j N_I d\Gamma_h \\ &\quad + \frac{1}{2}\int_{\Gamma_h} (nnode-2)F_I^j n_j N_I N_I d\Gamma_h \end{aligned} \tag{3}$$

For an interior point, the boundary integrals can be dropped and the right-hand-side becomes

$$\begin{aligned} & RHS(I) \\ &= -\int_{\Omega_h} (F^j + (nnode-2)F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h \\ &= -\sum_{e\in I}\int_{\Omega_e} (F^j + (nnode-2)F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h \\ &= -\sum_{e\in I}\int_{\Omega_e} (\sum_{J=1}^{nnode} N_J F_J^j + (nnode-2)F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h \\ &= -\sum_{e\in I}\int_{\Omega_e} (\sum_{\substack{J=1 \\ J\neq I}}^{nnode} N_J F_J^j + (nnode-1)F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h \\ &= -\sum_{e\in I}\int_{\Omega_e} \sum_{\substack{J=1 \\ J\neq I}}^{nnode} (N_J F_J^j + F_I^j N_I)\frac{\partial N_I}{\partial x_j} d\Omega_h. \end{aligned} \tag{4}$$

Note that

$$\int_{\Omega_e} N_J \frac{\partial N_I}{\partial x_j} d\Omega_h = \int_{\Omega_e} N_I \frac{\partial N_I}{\partial x_j} d\Omega_h. \tag{5}$$

Then the following expression is obtained

$$\begin{aligned} & RHS(I) \\ &= -\sum_{\substack{J=1 \\ J\neq I}}^{nnode} (\sum_{e\in I}\int_{\Omega_e} N_J \frac{\partial N_I}{\partial x_j} d\Omega_h)(F_J^j + F_I^j), \end{aligned} \tag{6}$$

which can be further simplified as

$$RHS(I) = \sum_{IJ}^{m_I} C_{IJ}^j (F_I^j + F_J^j), \tag{7}$$

where m_I is the number of edges connected to the node I, and

$$C_{IJ}^j = -\sum_{e\in IJ} \frac{\Omega_e}{4}\frac{\partial N_I}{\partial x_j}\Big|_e \ , \tag{8}$$

in 3D.

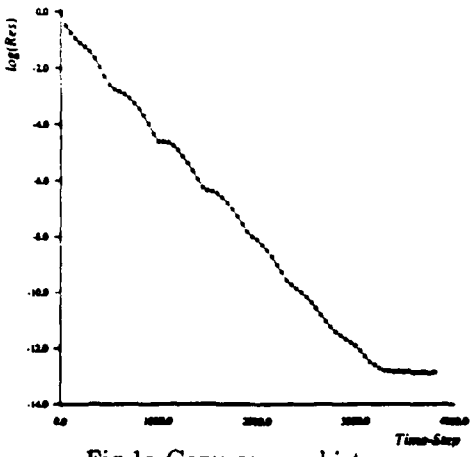


Fig.1a Convergence history

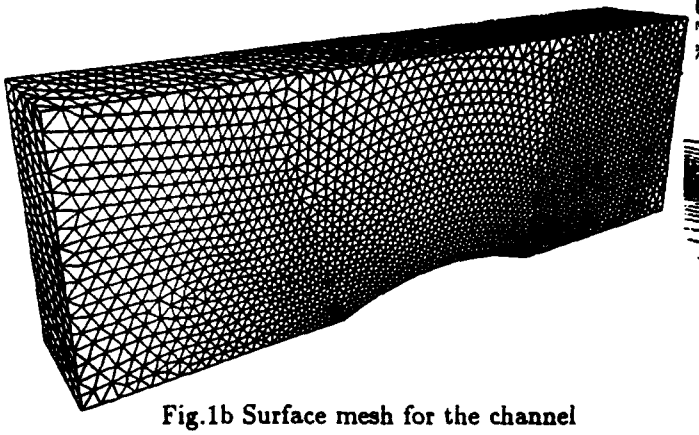


Fig.1b Surface mesh for the channel

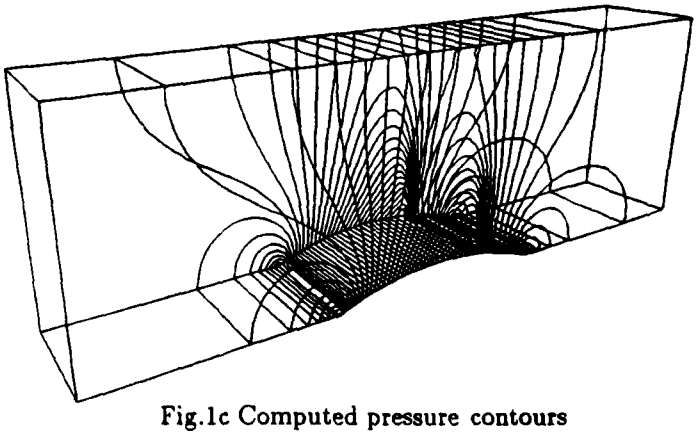


Fig.1c Computed pressure contours

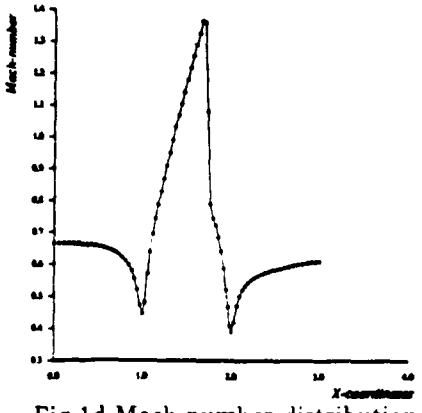


Fig.1d Mach number distribution

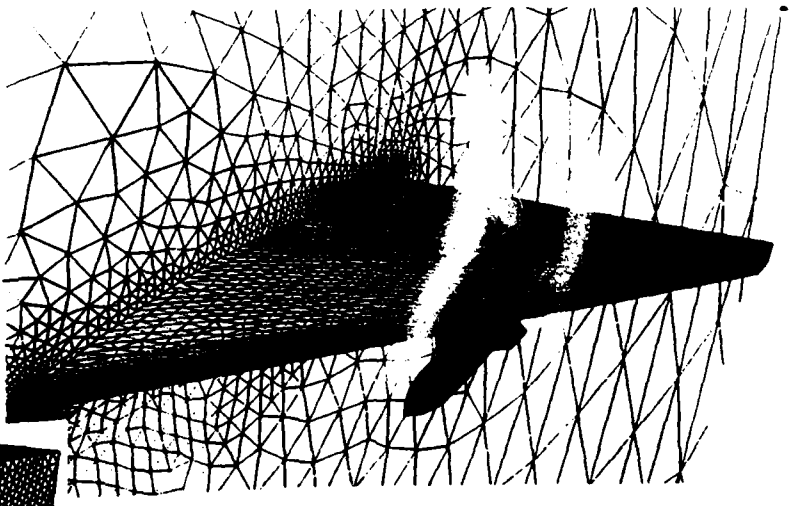


Fig.2a Surface mesh for the wing/pylon/store
nelem=1,518,770,npoin=274.953,nboun=33.046

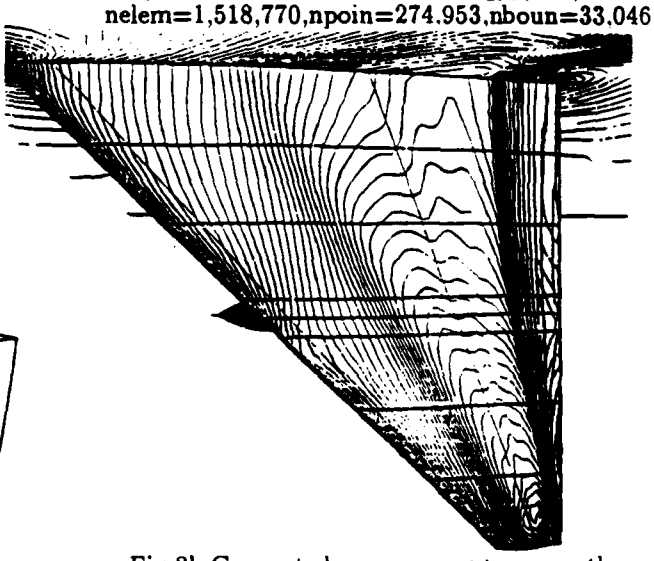


Fig.2b Computed pressure contours on the upper surface at $M_\infty = 0.95, \alpha = 0°$

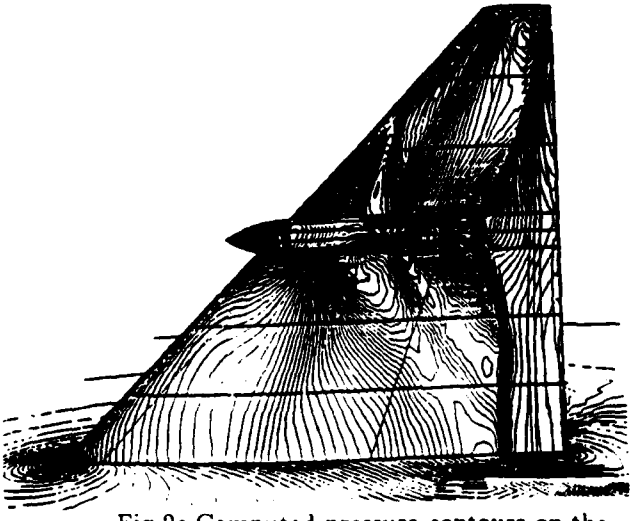


Fig.2c Computed pressure contours on the lower surface at $M_\infty = 0.95, \alpha = 0°$

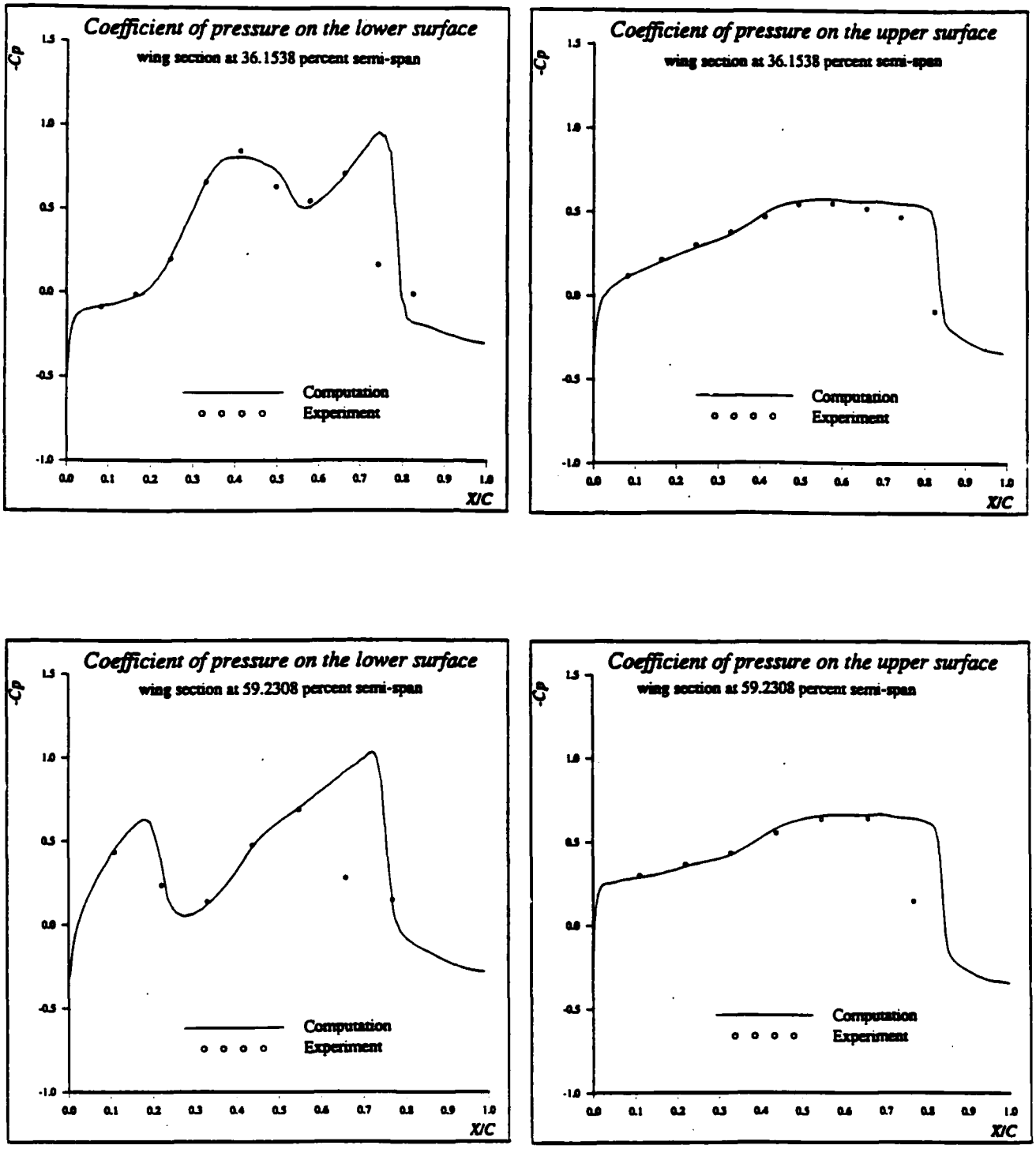


Fig.2d. Comparisons between computed and experimental surface pressure coefficient for the wing/pylon/store configuration $M_\infty = 0.95, \alpha = 0°$

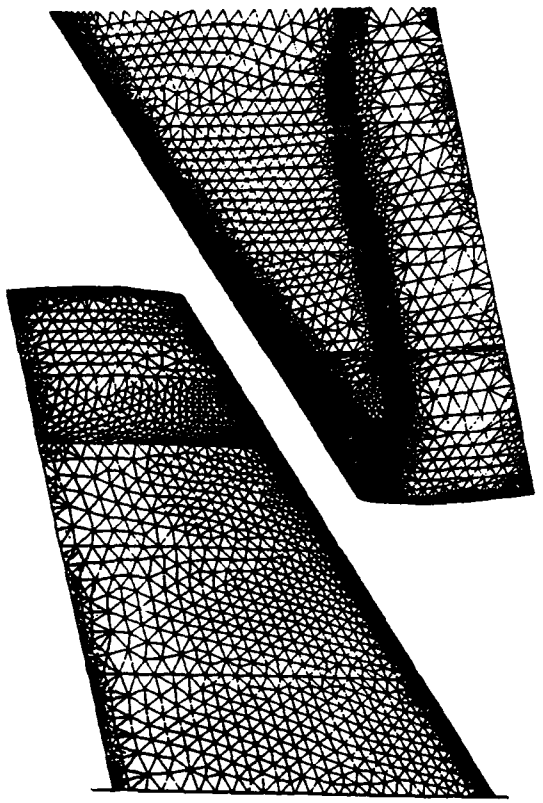


Fig.3a Upper and lower surface meshes for M6 wing; nelem=738,669,npoin=133,206,nboun=17,155

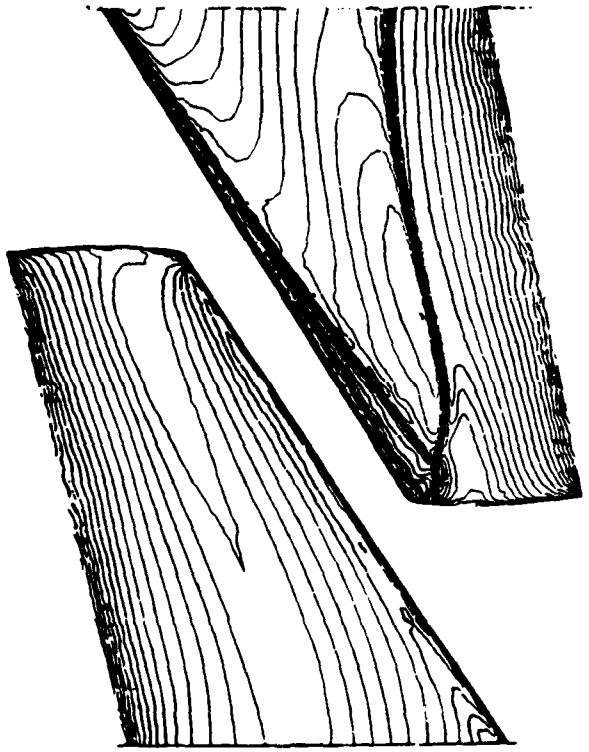


Fig.3b Computed pressure contours on the upper and lower surfaces; $M_\infty = 0.84, \alpha = 3.06°$

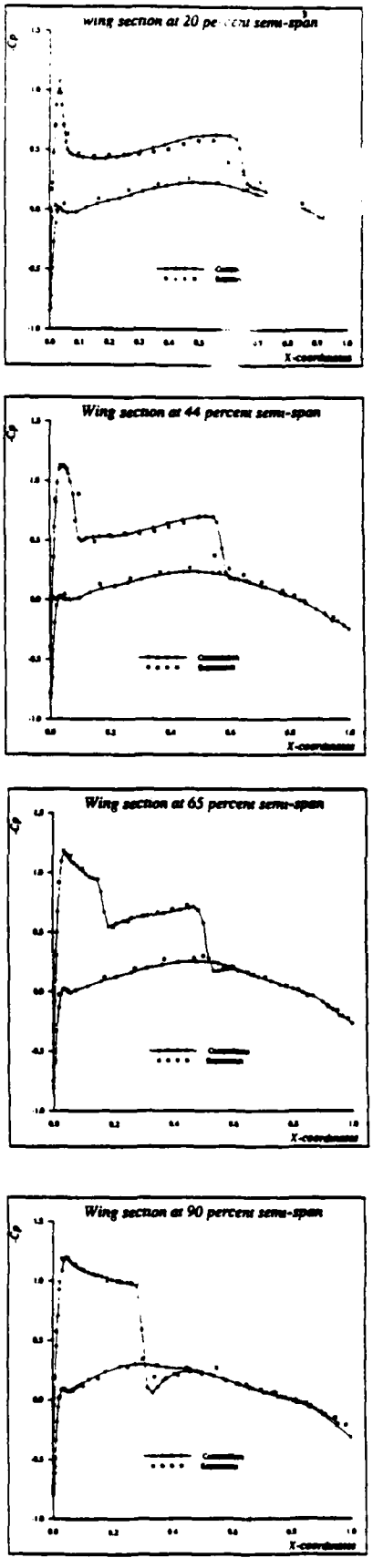


Fig.3c Compparison between computed and experimental surface pressure coefficient for M6wing

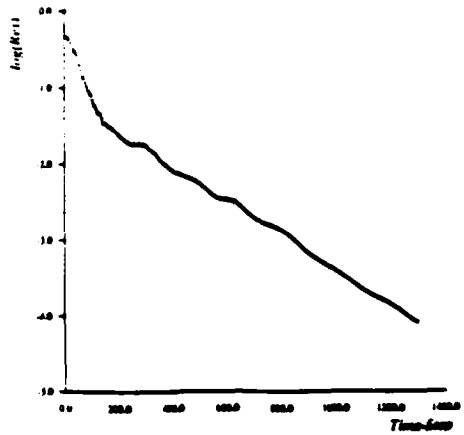


Fig.4a Convergence History for the F-18 Fighter

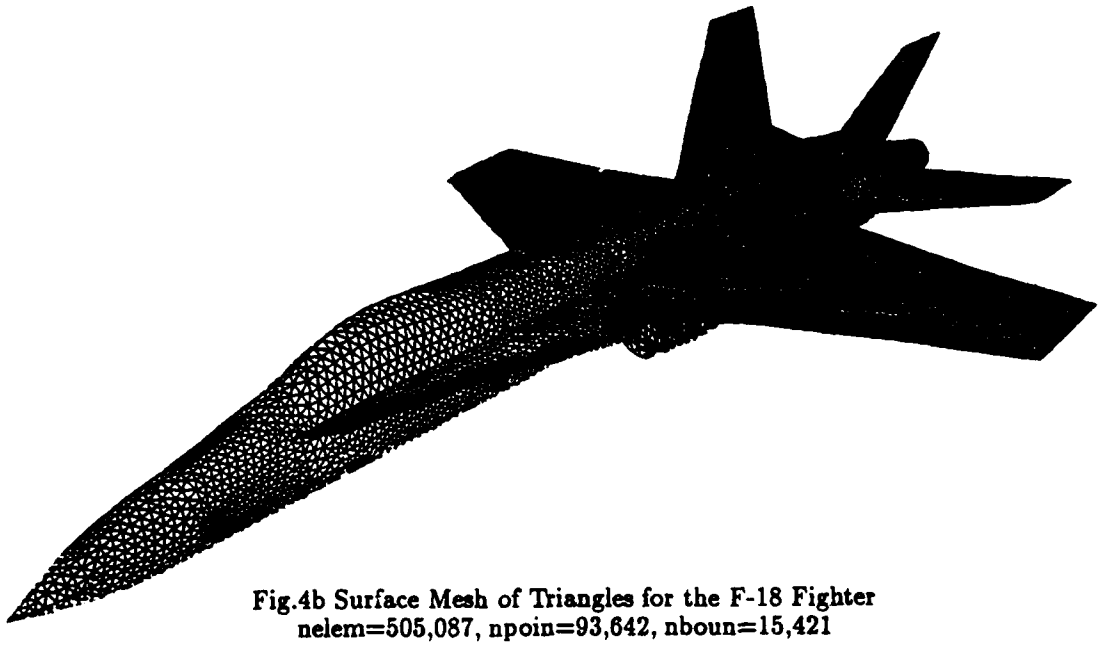


Fig.4b Surface Mesh of Triangles for the F-18 Fighter
nelem=505,087, npoin=93,642, nboun=15,421

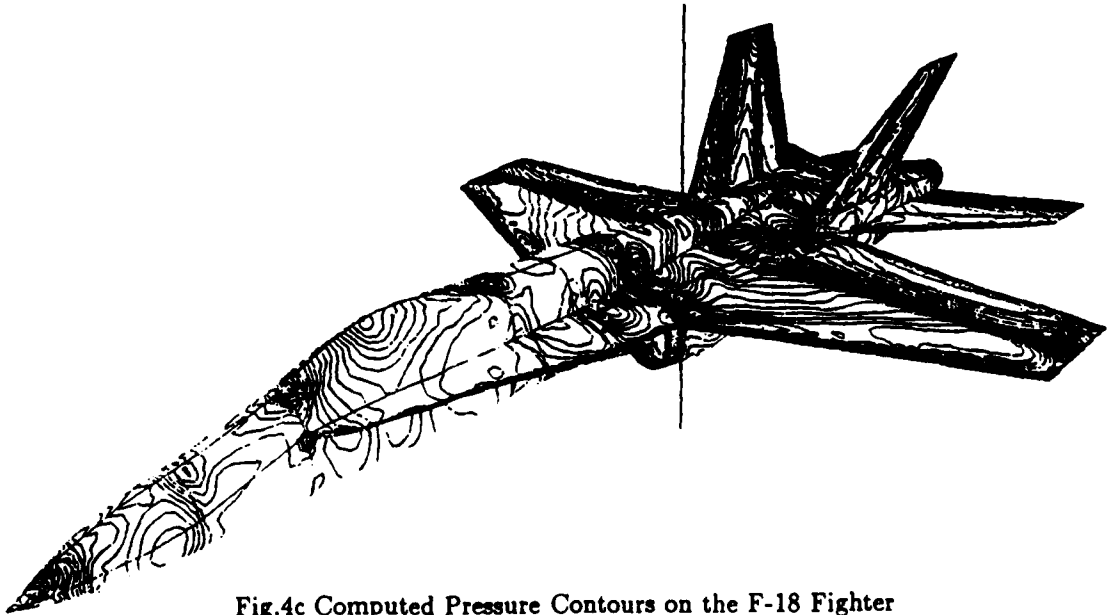


Fig.4c Computed Pressure Contours on the F-18 Fighter
at $M_\infty = 0.9, \alpha = 3°$

APPENDIX 4: SAMPLE RUN